Double patenting

WEST

3

Generate Collection Print

L14: Entry 1 of 22 File: USPT Sep 17, 2002

US-PAT-NO: 6453376
DOCUMENT-IDENTIFIER: US 6453376 B1

TITLE: Method for implementing scheduling mechanisms with selectable resource modes

DATE-ISSUED: September 17, 2002

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Fairman; Bruce A.	Woodside	CA		
Smyers; Scott D.	San Jose	CA		
Ludtke; Harold A.	San Jose	CA		
Stone; Glen D.	Campbell	CA		

US-CL-CURRENT: 710/240; 709/223, 709/224, 709/226, 710/107, 710/113, 710/119

CLAIMS:

What is claimed is:

1. A system for selectively allocating resources in an electronic device, comprising: a resource characterization set coupled to said electronic device, said resource characterization set corresponding to a requested process, said requested process including one or more time-sensitive isochronous processes for manipulating time-critical isochronous data; a software module configured to generate a process request to instantiate said requested process on said electronic device, said process request including an identifier that corresponds to a specific resource characterization from said resource characterization set; an allocation manager configured to handle said requested process by referencing said resource characterization set, said resource characterization set including a plurality of resource characterizations that each include resource requirements for executing said requested process, said plurality of resource characterizations including a most mode, a best mode, and a worst mode, said allocation manager selecting said resource requirements from one or more of said most mode, said best mode, and said worst mode, and comparing said resource requirements to currently-available resources, said allocation manager thereby authorizing said requested process only when said resource requirements are less than, or equal to, said currently-available resources; and a processor coupled to said electronic device for controlling said allocation manager.

2. The system of claim 1, wherein said electronic device is coupled to an electronic network that is implemented according to an IEEE Std 1394 serial bus interconnectivity standard.

3. The system of claim 1 wherein said electronic device is one of a consumer-electronics device, an audio-visual device, a set-top box, and a personal computer device.

4. The system of claim 1 wherein said allocation manager evaluates a resource characterization from said resource characterization set in response to said process request from said software module.

5. The system of claim 4 wherein said resource characterization includes one or more resource listings and one or more corresponding resource usage values that are required for executing said requested process.

6. The system of claim 4 wherein said resource characterization includes a most mode that provides an optimal performance quality for said requested process and utilizes a substantial amount of said resources.

7. The system of claim 4 wherein said resource characterization includes a best mode that provides an average performance quality for said requested process and utilizes a moderate amount of said resources.

8. The system of claim 4 wherein said resource characterization includes a worst mode that provides a minimum acceptable performance quality for said requested process and utilizes a reduced amount of said resources.

9. The system of claim 4 wherein said allocation manager compares resource usage values from said resource characterization and currently-available resource values to determine whether to authorize said requested process.

10. A system for selectively allocating resources in an electronic device, comprising: a resource characterization set coupled to said electronic device, said resource characterization set corresponding to a requested process; an allocation manager configured to handle said requested process by referencing said resource characterization set; and a processor coupled to said electronic device for controlling said allocation manager, wherein a software module generates a process request to instantiate said requested process on said electronic device, wherein said allocation manager evaluates a resource characterization from said resource characterization set in response to said process request from said software module, wherein said allocation manager compares resource usage values from said resource characterization and currently-available resource values to determine whether to authorize said requested process, and wherein said allocation manager authorizes said requested process whenever said resource usage values from said resource characterization are less than or equal to said currently-available resource values.

11. A system for selectively allocating resources in an electronic device, comprising: a resource characterization set coupled to said electronic device, said resource characterization set corresponding to a requested process; an allocation manager configured to handle said requested process by referencing said resource characterization set; and a processor coupled to said electronic device for controlling said allocation manager, wherein a software module generates a process request to instantiate said requested process on said electronic device, wherein said allocation manager evaluates a resource characterization from said resource characterization set in response to said process request from said software module, wherein said allocation manager compares resource usage values from said resource characterization and currently-available resource values to determine whether to authorize said requested process, and wherein said allocation manager denies said requested process whenever said resource usage values from said resource characterization are greater than said currently-available resource values.

12. The system of claim 10 wherein said allocation manager updates said currently-available resource values with said resource usage values whenever said requested process is authorized by said allocation manager.

13. The system of claim 10 wherein a picokernel in said electronic device instantiates and executes said requested process after said allocation manager authorizes said requested process.

14. A system for selectively allocating resources in an electronic device, comprising: a resource characterization set coupled to said electronic device, said resource characterization set corresponding to a requested process; an

allocation manager configured to handle said requested process by referencing said resource characterization set; and a processor coupled to said electronic device for controlling said allocation manager, wherein a software module generates a process request to instantiate said requested process on said electronic device, wherein said allocation manager evaluates a resource characterization from said resource characterization set in response to said process request from said software module, and wherein said allocation manager evaluates a most mode of said resource characterization set, and authorizes said requested process whenever said resource usage values from said most mode are less than or equal to currently-available resource values.

15. The system of claim 14 wherein said allocation manager evaluates a best mode of said resource characterization set whenever said resource usage values from said most mode are greater than said currently-available resource values, said allocation manager authorizing said requested process whenever said resource usage values from said best mode are less than or equal to said currently-available resource values.

16. The system of claim 15 wherein said allocation manager evaluates a worst mode of said resource characterization set whenever said resource usage values from said best mode are greater than said currently-available resource values, said allocation manager authorizing said requested process whenever said resource usage values from said worst mode are less than or equal to said currently-available resource values.

17. A method for selectively allocating resources in an electronic device, comprising the steps of: generating a process request with a software module to instantiate a requested process on said electronic device, said process request including an identifier that corresponds to a specific resource characterization from a resource characterization set; referencing said resource characterization set with an allocation manager, said resource characterization set corresponding to said requested process, said requested process including one or more time-sensitive isochronous processes for manipulating time-critical isochronous data; handling said requested process with said allocation manager based upon said resource characterization set, said resource characterization set including a plurality of resource characterizations that each include resource requirements for executing said requested process, said plurality of resource characterizations including a most mode, a best mode, and a worst mode, said allocation manager selecting said resource requirements from one or more of said most mode, said best mode, and said worst mode, and comparing said resource requirements to currently-available resources, said allocation manager thereby authorizing said requested process only when said resource requirements are less than, or equal to, said currently-available resources; and controlling said allocation manager with a processor that is coupled to said electronic device.

18. The method of claim 17, wherein said electronic device is coupled to an electronic network that is implemented according to an IEEE Std 1394 serial bus interconnectivity standard.

19. The method of claim 17 wherein said electronic device is one of a consumer-electronics device, an audio-visual device, a set-top box, and a personal computer device.

20. The method of claim 17 wherein said allocation manager evaluates a resource characterization from said resource characterization set in response to said process request from said software module.

21. The method of claim 20 wherein said resource characterization includes one or more resource listings and one or more corresponding resource usage values that are required for executing said requested process.

22. The method of claim 20 wherein said resource characterization includes a most mode that provides an optimal performance quality for said requested

process and utilizes a substantial amount of said resources.

23. The method of claim 20 wherein said resource characterization includes a best mode that provides an average performance quality for said requested process and utilizes a moderate amount of said resources.

24. The method of claim 20 wherein said resource characterization includes a worst mode that provides a minimum acceptable performance quality for said requested process and utilizes a reduced amount of said resources.

25. The method of claim 20 wherein said allocation manager compares resource usage values from said resource characterization and currently-available resource values to determine whether to authorize said requested process.

26. A method for selectively allocating resources in an electronic device, comprising: referencing a resource characterization set with an allocation manager, said resource characterization set corresponding to a requested process; handling said requested process with said allocation manager based upon said resource characterization set; and controlling said allocation manager with a processor that is coupled to said electronic device, wherein a software module generates a process request to instantiate said requested process on said electronic device, wherein said allocation manager evaluates a resource characterization from said resource characterization set in response to said process request from said software module, wherein said allocation manager compares resource usage values from said resource characterization and currently-available resource values to determine whether to authorize said requested process, and wherein said allocation manager authorizes said requested process whenever said resource usage values from said resource characterization are less than or equal to said currently-available resource values.

27. A method for selectively allocating resources in an electronic device comprising: referencing a resource characterization set with an allocation manager, said resource characterization set corresponding to a requested process; handling said requested process with said allocation manager based upon said resource characterization set; and controlling said allocation manager with a processor that is coupled to said electronic device, wherein a software module generates a process request to instantiate said requested process on said electronic device, wherein said allocation manager evaluates a resource characterization from said resource characterization set in response to said process request from said software module, wherein said allocation manager compares resource usage values from said resource characterization and currently-available resource values to determine whether to authorize said requested process, and wherein said allocation manager denies said requested process whenever said resource usage values from said resource characterization are greater than said currently-available resource values.

28. The method of claim 26 wherein said allocation manager updates said currently-available resource values with said resource usage values whenever said requested process is authorized by said allocation manager.

29. The method of claim 26 wherein a picokernel in said electronic device instantiates and executes said requested process after said allocation manager authorizes said requested process.

30. A method for selectively allocating resources in an electronic device, comprising: referencing a resource characterization set with an allocation manager, said resource characterization set corresponding to a requested process; handling said requested process with said allocation manager based upon said resource characterization set; and controlling said allocation manager with a processor that is coupled to said electronic device, wherein a software module generates a process request to instantiate said requested process on said electronic device, wherein said allocation manager evaluates a resource characterization from said resource characterization set in response

to said process request from said software module, and wherein said allocation manager evaluates a most mode of said resource characterization set, and authorizes said requested process whenever said resource usage values from said most mode are less than or equal to currently-available resource values.

31. The method of claim 30 wherein said allocation manager evaluates a best mode of said resource characterization set whenever said resource usage values from said most mode are greater than said currently-available resource values, said allocation manager authorizing said requested process whenever said resource usage values from said best mode are less than or equal to said currently-available resource values.

32. The method of claim 31 wherein said allocation manager evaluates a worst mode of said resource characterization set whenever said resource usage values from said best mode are greater than said currently-available resource values, said allocation manager authorizing said requested process whenever said resource usage values from said worst mode are less than or equal to said currently-available resource values.

33. A computer-readable medium comprising program instructions for selectively allocating resources in an electronic device by performing the steps of: generating a process request with a software module to instantiate a requested process on said electronic device, said process request including an identifier that corresponds to a specific resource characterization from a resource characterization set; referencing said resource characterization set with an allocation manager, said resource characterization set corresponding to said requested process, said requested process including one or more time-sensitive isochronous processes for manipulating time-critical isochronous data; handling said requested process with said allocation manager based upon said resource characterization set, said resource characterization set including a plurality of resource characterizations that each include resource requirements for executing said requested process, said plurality of resource characterizations including a most mode, a best mode, and a worst mode, said allocation manager selecting said resource requirements from one or more of said most mode, said best mode, and said worst mode, and comparing said resource requirements to currently-available resources, said allocation manager thereby authorizing said requested process only when said resource requirements are less than, or equal to, said currently-available resources; and controlling said allocation manager with a processor that is coupled to said electronic device.

34. A system for selectively allocating resources in an electronic device, comprising: means for referencing a resource characterization set that corresponds to a requested process; means for handling said requested process based upon said resource characterization set; and means for controlling said means for referencing and said means for handling.

WEST

Help | Logout | Interrupt

Main Menu | Search Form | Posting Counts | Show S Numbers | Edit S Numbers | Preferences | Cases

Search Results -

Term	Documents
IEEE.USPT.	110651
IEEES.USPT.	2
SERIAL.USPT.	141525
SERIALS.USPT.	104
BUS.USPT.	150223
BUSES.USPT.	38227
BUSSES.USPT.	13154
((IEEE NEAR5 SERIAL) ADJ BUS).USPT.	353
(IEEE NEAR5 SERIAL BUS).USPT.	353

Database:
- US Patents Full-Text Database
- US Pre-Grant Publication Full-Text Database
- JPO Abstracts Database
- EPO Abstracts Database
- Derwent World Patents Index
- IBM Technical Disclosure Bulletins

Search: Refine Search

Recall Text | Clear

Search History

DATE: Wednesday, October 16, 2002 Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
DB=USPT; PLUR=YES; OP=ADJ			
L28	IEEE near5 serial bus	353	L28
L27	IEEE Std near5 serial bus	5	L27
L26	IEEE STD near5 serial bus	5	L26
L25	L21 and l8	0	L25
L24	L19 and l8	0	L24
L23	l21 and l8	0	L23
L22	l17 and l8	8	L22
L21	l19 and l1.ab.	17	L21
L20	L19 and l2	2	L20
L19	L18 and optim$	31	L19
L18	L17 same l15	43	L18
L17	available resource$1	2091	L17
L16	available resorce$1	0	L16
L15	resource near1 requirement$1	905	L15
L14	L13 and l12 and l4	22	L14
L13	resource$1 near3 usage	1329	L13
L12	resource$1 near3 list$	1299	L12
L11	resource$1 neaer3 list$	0	L11
L10	L8 and l4	0	L10
L9	L8 and l6	0	L9
L8	IEEE standard and serial bus	365	L8
L7	L6 and l5	4	L7
L6	L2 same optim$	70	L6
L5	L4 and l1.ab.	33	L5
L4	l1 and l2 and l3	137	L4
L3	L2 and optim$	370	L3
L2	resource$1 near5 minimum	1212	L2
L1	resource$1 near5 allocat$	7215	L1

END OF SEARCH HISTORY

WEST

End of Result Set

☐ Generate Collection Print

L20: Entry 2 of 2 File: USPT Apr 18, 1995

DOCUMENT-IDENTIFIER: US 5408663 A
TITLE: Resource allocation methods

Abstract Text (1):
Methods of operating a digital computer to optimize project scheduling. Where the overall effects of a schedule, such as total project duration or cost, are unsatisfactory, the schedule is processed iteratively so that on each iteration a particular task is selected for modification according to a preset policy and data defining an aspect of that task is adjusted in a small step. A schedule is further optimized to fit the available resources by a repetitive process of assigning resources having the proper capabilities to tasks according to a predetermined order of tasks and testing whether the assigned resource can permit shortening of the task duration. Further methods select an optimum mix of capabilities to be provided by each of several resources to be hired for a project.

Brief Summary Text (4):
Ordinarily, each schedule includes one or more critical paths. A critical path in this context can be thought of as a series of tasks forming the longest duration path between the start and the end of the project. Such critical path has duration equal to the duration of the schedule as a whole. Non-critical paths are a series of tasks having aggregate duration less than the aggregate duration of the critical path. The non-critical paths thus contain some "slack" time, equal to the difference between the duration of the critical path and the duration of the non-critical path. The input data provided to the computing system may also include resource requirements for the various tasks as, for example, the number of workers having particular skills or machines of a certain type required to perform each particular task. The input data may further include data defining the maximum available resources, such as the number of workers and machines available to the project. In common scheduling methods, automatic data processing equipment tests the schedule to determine whether the sum of resources required by all tasks operating at a given time exceeds the sum of the resources available. If so, the system uses a common algorithm referred to as "levelling" to adjust the starting and ending times of various tasks until the maximum resource demand for each time is less than the maximum resource availability.

Brief Summary Text (5):
The conventional methods described above provide a shortest-duration schedule for the particular set of assumed task duration and resource availabilities supplied to the process as input data. However, the known process has essentially no capability for adjusting a schedule to meet external constraints or to compensate for external resource availability changes. For example, if the schedule calculated by these methods yields a total project duration greater than the maximum duration allowable according to external constraints, the system provides essentially no guidance as to how one might best reduce the project duration while maintaining other desired attributes such as risk and cost. For simple projects, schedulers have used "brute force" or guesswork techniques to arrive at the modified schedules which would meet desired output targets as, for example, a maximum duration. Thus, a schedule may simply cut the durations of all tasks on the critical path by a fixed percentage to meet a maximum duration. As a general rule results achieved by such guesswork processes are decidedly suboptimal. For example, where durations of various tasks are adjusted to make the entire schedule conform to a maximum duration, the costs and risks associated with the schedule tended to increase substantially. With current methods, these increases may be far greater than if an optimum strategy were employed. Stated another way, it may be possible to achieve the same reduction in

duration with less of an increase in a cost and risk.

Brief Summary Text (6):
Moreover, methods of fitting available resources tasks at hand have been sub-optimal. The so-called levelling methods merely alleviate conflicts or overuse of the resources. They offer no answer to the question of whether the project schedule could be further optimized through a better use of available resources. Moreover, these methods can produce irrational results, such as resource allocations which require programmers to switch projects every few hours.

Brief Summary Text (7):
The inability of the prior art scheduling methods to provide optimum schedules for meeting external criteria or to provide optimum deployment of available resources to enhance a schedule have imposed substantial unnecessary costs in numerous different industries. Nonetheless, the substantial need for improved scheduling and resource allocation methods and apparatus has not been fulfilled to date.

Brief Summary Text (14):
The preferred methods according to this aspect of the invention incorporate the discovery that step wise adjustments of the input data associated with a selected task in an iterative process, together with renewed selection of the task to be so modified after each iteration, will yield a schedule which is optimum, or at least far closer to optimum than would be achieved by arbitrary selection of a single task or group of tasks for input data modification in a single step up. Merely by way of example, in a case where the duration estimated for a particular task reduced step wise, that step wise reduction may alter the schedule so that the just-reduced task is no longer on the critical path. Accordingly, any further adjustments should be made in other tasks. Further, even if the newly modified task remains on the critical path, the just-completed modification may alter the relationship of the modified task to the task selection criteria, and therefore may make it preferable to select other tasks for further modification. For example, where the modification policy involves selection of the task on the critical path with the lowest risk of delayed completion and reduction of the duration estimate for that task, a particular step wise reduction in a duration estimate may cause the risk of delayed completion for that task to rise so that on any subsequent repetition a different task should be modified. Stated another way, the step wise, gradual modification of input data for the various tasks, with repeated reapplication of the selection and modification policy assures that these policies are applied optimally until the schedule is revised to meet the desired effects criteria. Methods according to this aspect of the invention thus can provide schedules which are closer to optimum, and provide a better combination of desirable global effects parameters than conventional methods employing arbitrary selection of tasks for modification and attempts to modify the task or tasks so selected in a single step from the initial schedule.

Brief Summary Text (16):
A further aspect of the present invention provides methods of optimizing a schedule with respect to available resources. The schedule being optimized is referred to as the "working" schedule and is defined by schedule data. The schedule data includes data representing a plurality of tasks, earliest start and latest end times associated with each task and a subset of tasks representing a critical path. The data provided further includes resource data representing a set of capabilities possessed by each resource and quantities of individual resources available as a function of time, together with a set of capability requirements and quantity requirements for each task. As used herein, the term "capability" refers to a capability applicable to a task. Thus, a capability of a worker may be "UNIX programming" or "jet engine mechanic" whereas a capability of a physical machine may be expressed as "injection molding press time 560 tons" or the like. Each resource may include multiple capabilities as, for example, a particular worker who is both a UNIX programmer and a systems programmer. The quantities of resources may be expressed as such, i.e., X man-hours of the aforementioned worker available on this day and Y man-hours of the same worker available on another day. The capability requirements and quantity requirements for the tasks are set with reference to particular capabilities, and not with reference to particular resources. Thus, the requirements for a particular task might specify "UNIX programmer 8 man-hours" and "Systems Programmer 2 man-hours" and could be satisfied by any worker possessing the requisite capabilities.

Brief Summary Text (18):

The method desirably further includes the step of testing each task on the critical path of the working schedule to determine whether or not the resources allocated to the task would permit completion of the task in a shorter duration than the duration assigned to the task in the working schedule and, if so, generating a positive test signal. The method desirably includes the further step, performed only if a positive test signal is generated for at least one task in the working schedule, of computing a new working schedule by resetting the duration required for each task based upon the resources allocated to that task and computing the earliest start and latest end times for all tasks in the schedule based upon such duration. Desirably, all of the aforementioned steps are repeated so that new working schedules are generated iteratively and subjected to resource allocation and testing as aforementioned. This process continues until no positive-test signal is generated, thus indicating that the last computed working schedule is substantially an optimum schedule.

Brief Summary Text (21):
Most preferably, the two aspects of the invention discussed above--that of allocating time and resources by modifying a schedule to meet a set of effects criteria, and that of optimizing the schedule with respect to available resources are combined. Thus, the step of optimizing by fitting resources and capabilities to tasks may be performed for the first found schedule in the iterative step of revising a schedule to meet effects criteria, or, for the last schedule in that process, or both. It is ordinarily unnecessary to perform this resource fitting method for all steps of the process.

Brief Summary Text (22):
Yet another aspect of the present invention provides ways to determine optimum resource sets for performing a particular task. This aspect of the present invention provides methods of operating a digital computer to generate output signals representing an optimum set of resources for performing a task according to a particular schedule. That is, where a given resource, typically a person, can have more than one capability, and where only a finite number of resources can be employed for the entire project, what capability should each resource have? For example, in a project involving four different computer languages and only two programmers, is it better to hire a first programmer with basic and C++ programming capabilities and a second with FORTRAN and SQL, or to hire a first programmer with basic and SQL capabilities and a second with FORTRAN and C++ capability. With a small schedule, the answer can be determined by inspection and intuition; if the tasks involving C++ programming are scheduled to begin and end at the same times as the tasks involving basic programming, then it is immediately apparent that the basic and C++ capability should be provided by two different people, and not by the same person because that person would be overloaded. However, where the project involves many people and many tasks, inspection and intuition do not suffice.

Brief Summary Text (23):
The methods according to this aspect of the invention desirably include the steps of providing resource constraint data defining the number of resources to be included in the optimum set and providing working schedule data representing the plural tasks in the schedule. The working schedule data incorporates earliest starting and latest-ending times for each task; these times constitute the event horizon for the task. The method also includes the steps of providing capability and quantity requirement data for each task defining the identity of each capability required for the task and the quantity or effort requirement for that capability, as, for example, in man-hours. The capabilities identified in the capability requirements data for all of the tasks are ranked according to a preselected capability ranking scheme. The capability ranking scheme may be arranged to give highest rank to those capabilities required in the largest amount in the schedule as a whole. The method further includes the step of setting an order of tasks for resource allocation, most preferably with those tasks having the earliest latest ending time or T.sub.l first in such order.

Brief Summary Text (27):
The system then repeats the task selection steps and the subsequent steps using the same selected capability until all of the tasks have been compared with respect to the selected capability and until all tasks calling for such capability have been processed and allocated time from the resources. Following that, the system selects a new capability and repeats again until all of the capabilities have been used, i.e., until all capabilities have been compared with all tasks and allocated as discussed above. Once this process is completed, the contents of the resource data table are output from the computer as an output signal. The resource data table

includes data defining the capabilities of the resources needed to perform the tasks of the schedule, optimized in accordance with the resource selection policy. In real terms, the output signal will include data specifying which capabilities the various people to be hired for a job should have so as to perform the tasks in the schedule most efficiently. The composition of resources will depend upon the selection policy employed with a "concentrating" policy. The system effectively assures that the resources having larger numbers of already assigned capabilities will be assigned further capabilities and hence that all of the capabilities required to perform the task will be assigned to the minimum number of resources. The output signal will call for a set of resources, having the required capabilities clustered on the smallest number of persons consistent with the schedule. Conversely, if the selection policy is a "spreading" policy which assigns additional capabilities to those resources having the fewest already assigned capabilities, the resulting output signal will specify a set of resources having the various capabilities spread among the maximum number of people. Preferred systems according to this aspect of the present invention provide a way to specify the best set of abilities or skills in the personnel to be hired for a project.

Detailed Description Text (6):
The computer is further supplied with policy data defining task selection policies and task adjustment policies. As further discussed below, the task selection policy may include a minimum risk selection policy or a minimum resource impact selection policy, whereas the task modification policy may include either a modify-cost policy or a modify-time policy.

Detailed Description Text (21):
In successive repetitions if the system encounters a new task with a lower level of risk, the previously selected task will be replaced by the new task. Thus, the system will select the task with the lowest level of risk before passing out of the loop at step 56. Alternatively, if the policy data supplied at step 22 was for a minimum resource impact policy, the system would not ever pass through step 58. Instead, for each critical-path task, the system would branch from step 57 to step 66. In step 66, the system tests the duration of the current task against the maximum task duration previously recorded for previously processed tasks. If the current task duration is no greater than the maximum previously set, the system passes to step 56 and repeats the loop. If the current task has a duration greater than the previously recorded maximum, the maximum is reset to the duration of the current task in step 68 and the current task is marked as the selected task in step 70. Thus, in a manner analogous to the selection of tasks for risks, this branch of the program will select the particular task on the critical path of the schedule having the longest duration of all tasks on the critical path. Other, similar branches (not shown) can be interposed between step 57 and step 56 for other tasks selection policies which may be specified in the policy data of step 22. For example, another policy would select tasks by giving some resources a preference over others, for example, by determining which task has the lowest requirement for the capability provided by a known scarce resource. Thus, a branch between steps 56 and 57 would calculate the amount of the particular capability in question required by the task and set the task on the current pass as the selected task if the amount for that task is less than the current minimum. Yet another policy which could be implemented by another branch is to select the task which most likely will have the lowest cost impact if modified.

Detailed Description Text (26):
As will be appreciated, the modification process is dynamic, and constantly readjusts itself during the process. Thus, each pass through the modification step 72 (FIG. 5) results in changes to the input data defining the various tasks. The effort or expected duration of at least some tasks may be modified. In a sense, the system determines the inputs required to get given output in the global effects vector of the schedule. Also, the particular tasks for which input data are modified may, and typically do, change on different passes through the selection step 52 and the modification step 72. As noted above, the calculation of a new schedule as in step 84 or step 78 (FIG. 5) may result in a new schedule with does not include the just-modified task on the critical path. Thus, on the next path through the BTTR or task selection step 52, that step will not be found on the critical path step 54 and hence will not be modified. Also, the task selection step 52 is applied in a dynamic manner, so as to select the task to modify in a new schedule based upon the considerations pertaining to that new schedule. For example, where the duration estimate of a particular task is reduced, the duration estimate will move closer to the best-case duration and further from the worse-case duration. If a risk-based

selection policy is applied, the risk calculated at step 58 for that particular task typically will be greater than the risk for the same task with a longer duration estimate. Stated another way, if the duration estimate for a task is reduced, there is a greater probability that the actual time to perform the task will be greater than the estimate. Thus, a task which was selected on one pass as having the lowest risk may be modified and may no longer have the lowest risk in the new schedule examined on the next path. The system tracks the most favorable modifications according to the preset task selection and modification policies dynamically through each pass. In effect, the system picks the best modification after each small stepwise modification has been completed. This results in a substantially optimum overall modification of the schedule, typically for superior to that which would be found by selection of one modification in a single step.

Detailed Description Text (27):
As will be further discussed below, the system does not take account of maximum resource availability data on each pass. Therefore, at this stage of the process, the schedule selected as acceptably meeting the effects criteria may in fact call for more resources than are available. Moreover, even if the schedule fits within the available resources, it may not be optimum for the available resources.

Detailed Description Text (34):
Thus, the system will converge to a particular schedule which best fits the available resources to the tasks. It can be shown mathematically that the schedule arrived at by allocating resources to tasks having the earliest T.sub.l first will be substantially an optimum schedule, that is, the strategy of fitting resources to tasks having the earliest T.sub.l first and then allocating resources to other tasks from the remaining resources is at least substantially isomorphic with any true optimization. Stated another way, given the available resources and a particular schedule supplied at the start of step 90, the schedule achieved at the end of step 90 (the "no" branch from step 122) is either an optimum fit of resources to tasks or so close as to be practically the same as an optimum.

Detailed Description Text (36):
When the system exits from the resource allocation stage 90, it tests to see if the overload signal has been set. If not, then the process is at an end. That is, the schedule has been adjusted to meet the effects requirements and the schedule has then been optimized to best fit with the available resources. However, if the overload signal is present at this stage, this indicates that the last-processed schedule encountered an overload condition with respect to at least one requirement of at least one process. This indicates that, even with the best possible allocation of resources to fit the schedule, there is at least one requirement for at least one task which cannot be met given the constraints imposed by the schedule. Such a condition most often will occur when the duration estimates for particular tasks are reduced during the input data adjustment stage 72 (FIGS. 2 and 5). In this condition, the system enters a reset and repeat mode 130. The reset and repeat mode may include the steps of providing an output overload description signal incorporating information that identifies the capability requirement and resource involved in each overload, the extent of the overload and the time when the overload occurs. This signal can be provided to an operator in perceptible form. This immediately informs the operator that, if he wishes to meet the constraints imposed by the effects requirements, he must provide the particular additional resources identified by the overload data. It is significant that the output overload description signal data accurately identifies what is needed to meet the schedule and hence what is needed to meet the effects requirements. Thus, the system not only tells the operator that additional resources are needed, but tells him which resources are needed to meet the requirements in an optimum manner. The system can be adjusted at will by the user to simulate the effects of different constraints on the system. For example, the user can repeat the process using different effects requirements to get an accurate answer as to what additional resources would be required to meet such requirements. Conversely, the user can simulate the effect of added resources on the ability to meet effects requirements. In each case, the system accurately converges to an optimum or essentially optimum schedule given the outside constraints.

Detailed Description Text (37):
Numerous variations and combinations of the features described above. In one such variation, the reset and repeat procedure 130 is arranged to operate automatically, so as to modify the input data automatically and add additional resources required by the overload signals. In a further variation, the resource allocation procedure

90 is performed immediately after the resource-levelling step 30 so that the effects vector is calculated after that first resource allocation stage. In this variant, the resource allocation step normally is not performed after each iteration of the task selection and input data adjustment steps 52 and 72 but rather is only repeated after an acceptable schedule has been computed using those steps. Also, the resource allocation step 90 may be applied to optimize the fit of available resources to any schedule, even if that schedule has not been computed according to the other steps of the process.

Detailed Description Text (38):
As illustrated in FIG. 7, a process 200 according to a further embodiment of the invention is arranged to determine what capabilities the resources used on a given project should have. In step 202 of the method, the computer is provided with resource constraint data, specifying the maximum number of resources which can be used and, optionally, also specifying a minimum number of resources, and also optionally specifying a maximum number of skills per resource. The computer is further provided with working schedule data in step 204. The working schedule data may be derived from a scheduling process as discussed above with reference to FIGS. 1-2, or else may be an arbitrary schedule. The working schedule data includes data defining the earliest starting and latest ending times, and hence the event horizon, for each task in the schedule and further includes capability and quantity requirement data. This data defines the specific capability or capabilities required for each task and the amount or quantity of effort required with respect to each such capability. Preferably, this data is in the form of task resource requirements or "TKRs" as discussed above. Each TKR, as discussed above specifies one capability and the quantity of effort, typically units of time, required in such capability for the particular task. At step 206, the computer initializes a resource data table with entries for a plurality of resources. Each entry includes resource schedule data in the form of a resource calendar as discussed above, defining the availability of the resource as a function of time. Each entry also includes resource capability data defining the capabilities assigned to the particular resource. In the method illustrated, the resource capability data is initially blanks. That is, the resource capability data lists no capabilities for any resource. The resource availability data normally shows all resources as fully available, i.e., with no portion of the resources time allocated. For example, in a typical project where each person assigned to the project is expected to work 40 hours per week, the initial resource availability data will show the person as available for time aggregating 40 hours per week.

Detailed Description Text (46):
During the repetitive cycling, capabilities are progressively assigned to the resources. When the system reaches the end of the process, the resource data table will specify a set of resources having the capabilities needed to perform the tasks of the project. If the overload flag was set in step 226 for any resource, and if that resource was modified to have a greater time availability than originally provided, this indicates that the tasks of the schedule cannot be met without overtime or expanded capability by the number of resources originally contemplated by the resource constraint data. If the concentration policy was in effect at step 224, the resource data table may include some resources with numerous capabilities and may also include some resources with null capabilities and full availability, i.e., some resources may not be allocated at all. This indicates that the schedule can be met using fewer resources then provided for in the constraint data. In effect, this determines the smallest number of people who can be assigned to a given project assuming that the people involved have the optimum set of capabilities for the project. Further, the output data specifies which capabilities should be provided by which individuals. The set of capabilities assigned to each resource can be used for example as a set of hiring specifications for staffing the project. The data generated using the concentrating policy typically will call for numerous capabilities in relatively few individuals. By contrast, the data generated using the "spreading" policies will generate hiring specifications that typically require relatively few capabilities in each individual but call for more individuals, up to the maximum specified by the resource constraint data.

CLAIMS:

4. A method as claimed in claim 1 further comprising the steps of providing resource data in said computer representing the quantities of individual resources available as a function of time and applicability data representing capabilities by each resource and a set of capability requirements for each task, and optimizing at least

one said schedule with respect to available resources by treating each said schedule as a working schedule to be optimized and:

(1) setting an order of tasks for resource allocation;

(2) defining a set of resources available according to said resource data for a task during the interval from the earliest starting time to the latest ending time as defined by the working schedule, such set including only resources having capabilities matching the capability requirements of the task;

(3) assigning resources from said set of resources to the task according to a preset resource selection scheme and modifying said resource data to indicate that the assigned resources are unavailable to other tasks;

(4) repeating said defining and assigning steps for all of the tasks in the working schedule in said order of tasks;

(5) testing each task on the critical path of the working schedule by determining whether the resources assigned to such task would permit completion of the task in an interval shorter than the earliest-start to latest-end interval associated with that task in the working schedule and, if so, generating a positive test signal; and

(6) if a positive-test signal is generated for any task in the working schedule, computing a new working schedule by setting the interval required for each task based upon the resources assigned to that task and computing earliest-start and latest-end times for all tasks based upon such intervals; and p1 (7) repeating steps (1)-(6) for each newly-computed working schedule until no positive-test signal is generated,

whereby the last-computed schedule is substantially an optimum schedule with respect to available resources.

5. A method as claimed in claim 4 further comprising the step of computing at least one said effect parameter in said computer based at least in part upon the quantities of resources assigned to said tasks in said optimizing process.

7. A method as claimed in claim 5 wherein said resource-optimizing step is performed in said computer for said initial set of schedule data prior to said comparing step.

10. A computer-implemented method of finding substantially optimal working schedule data with respect to available resources, said working schedule data including data representing a plurality of tasks, a set of capability requirements and quantity requirements for each task, earliest-start and latest-end times associated with each task and a subset of tasks representing a critical path, the method including the steps of providing, resource data representing a set of capabilities possessed by each resource and quantities of individual resources available as a function of time:

(1) setting an order of tasks for resource allocation;

(2) defining a set of resources available according to said resource data for a task during the interval from the earliest starting time to the latest ending time as defined by the working schedule, such set including only resources having capabilities matching the capability requirements of the task;

(3) assigning resources from said set of resources to the task according to a preset resource selection scheme and modifying said resource data to indicate that the assigned resources are unavailable to other tasks;

(4) repeating said defining and assigning steps for all of the tasks in the working schedule in said order of tasks;

(5) testing each task on the critical path of the working schedule by determining whether the resources assigned to such task would permit completion of the task in an interval shorter than the earliest-start to latest-end interval associated with the task in the working schedule and, if so, generating a positive test signal; and

(6) if a positive-test signal is generated for any task in the working schedule, computing a new working schedule by setting the interval required for that task based upon the resources assigned to that task and computing earliest-start and latest-end times for all tasks based upon such intervals; and

(7) repeating steps (1)-(6) for each newly-computed working schedule until no positive-test signal is generated,

whereby the last-computed working schedule is substantially an optimum schedule.

17. A method as claimed in claim 10 further comprising the step of altering said resource data to indicate availabilities of different quantities of resources and repeating said steps to thereby find an optimum schedule for each of a plurality of different resource availabilities.

20. A computer-implemented method of operating a digital computer to generate output signals representing a substantially optimum set of resources for performing a task according to a schedule comprising the steps of:

(A) providing resource constraint data defining the number of resources which can be included in the optimum set;

(B) providing working schedule data representing a plurality of tasks and representing earliest-starting and latest-ending times constituting the event horizon for each task and providing capability and quantity requirement data for each task defining the identity and quantity of each capability required for the task;

(C) ranking all of the capabilities identified in said capability requirement data with respect to all of said tasks according to a preselected capability ranking scheme;

(D) setting an order of tasks for resource allocation;

(E) providing a resource data table in said computer including resource identity data defining a plurality of resources; and resource schedule data defined the availability of each resource as a function of time, and resource capability data defining the capabilities assigned to each resource;

(F) selecting a capability according to said ranking of capabilities;

(G) using the selected capability to select a task by comparing data representing the selected capability with the capability requirement data of said tasks according to said order of tasks until a task is found having capability requirement data calling for the selected capability;

(H) processing the selected capability and task by:

(i) comparing the selected capability with the capability data in said resource data table to select a set of proper-capability resources for which the capabilities assigned to the resource include the selected capability;

(ii) comparing the availability of resources as a function of time to the event horizon for the selected task and the quantity requirement data for the task to select a set of adequate-availability resources for which the availability of the resource within the event horizon of the task is equal to or greater than the quantity requirement; and

(iii) determining whether any matching resource exists which is both a proper-capability resource and an adequate-availability resource;

(I) assigning a resource for time allocation by:

(i) if a matching resource is found in step (H)(iii), assigning said matching resource for time allocation; and

(ii) if no matching resource is found in step (H)(iii), applying a predetermined selection policy to select a resource, modifying the data in said resource data table with respect to said selected resource so that the selected resource is a

matching resource and assigning said selected resource for time allocation;

(J) allocating the time of the assigned resource to the selected task by modifying the availability data in said resource data table for the assigned resource to decrease the available time by an amount equal to the quantity requirement for the selected capability in the selected task during the event horizon of the selected task;

(K) using the same selected capability, repeating steps (G) through (J) until all tasks have been compared in step (G) and, if selected, processed through step (J);

(L) repeating steps (F) through (K) using a new capability on each repetition, until all of said capabilities have been used; and then

(K) outputting the contents of said resource data table as the output signal.

WEST

Generate Collection Print

L21: Entry 9 of 17 File: USPT Sep 15, 1998

DOCUMENT-IDENTIFIER: US 5809329 A
TITLE: System for managing the configuration of a computer system

Abstract Text (1):
A system for managing the configuration of devices of a computer system. Device information is obtained to uniquely identify each device and to describe the device characteristics associated with device operation. To obtain device information, a particular device is detected on a selected system bus and thereafter assigned an identification code that uniquely identifies the detected device. A system bus code, which uniquely identifies the selected system bus, is appended to the identification code, thereby forming a device identification code associated with the particular device. Logical configuration data, which supplies configuration requirements for device operation, is also obtained for the detected device. This data collection process is repeated until device information is obtained for each of the devices connected to the selected system bus. Resources are allocated to each device based on the device identification code and the logical configuration data. This resource allocation process prevents a potential conflicting use of the resources by the devices. A device driver, which enables communications between the corresponding device and the computer system, is identified and loaded for each of the devices in response to the device information. If the computer system contains more than one system bus, then the tasks of collecting device information, allocating resources, and identifying and loading device drivers are completed for each of the remaining system buses.

Brief Summary Text (12):
To overcome the frustration of users with present complicated configuration processes, it would be desirable to provide a system for automatically configuring a peripheral device or adapter board for a computer system. A system is needed to enable a user to simply connect a device to the computer, turn on the computer, and have the device properly work with the computer. There is a further need for a system that determines the optimal configuration for its resources and enables application programs to fully utilize the available resources.

Brief Summary Text (15):
The problems associated with the manual installation and configuration of adapter boards and peripheral devices for computer systems are solved by the principles of the present invention. The present invention provides a system for configuring the hardware and software components of a computer system by optimally allocating system resources for use by computer devices.

Brief Summary Text (55):
It is a further object of the present invention to provide a system that optimally allocates resources for use by the devices of the computer system.

Detailed Description Text (2):
To overcome the frustration of users with the present complicated and technical configuration processes for personal computers, the present invention provides a system for automatically configuring a peripheral device or an add-on type adapter board for use with a base or mobile computer system. The present invention enables a user to simply connect a new device to the computer, power the computer, and have the device properly work with the computer without user intervention. To provide this capability, the present invention determines the optimal configuration for the resources and enables the devices and the application programs to fully utilize the available resources. This can be accomplished for numerous computer bus architectures and types of devices.

Detailed Description Text (50):
The device information collected during the preboot operation further supports the allocation and the assignment of the resources 14 required by the nonboot-level devices on the integrated bus 15. In step 72, the resource requirements and dependencies for each of the nonboot-level devices 20 on the integrated system bus 15 are compared to the available resources 14. This comparison permits a determination of whether a potential resource conflict exists. In an iterative fashion, potential resource conflicts are arbitrated and resolved prior to resource allocation. In step 74, the resources 14 are allocated to the nonboot-level devices 20 based upon the arbitration results of the step 72 and those devices are configured in step 76. In view of the allocated resources, the identified device drivers are loaded in step 78 and the devices are enabled for operation with the computer 8. This arbitration process is described in more detail below with respect to FIG. 13.

WEST

End of Result Set

Generate Collection | Print

L7: Entry 4 of 4 File: USPT Dec 5, 1989

DOCUMENT-IDENTIFIER: US 4885686 A
TITLE: Methods and apparatus for efficient resource allocation

Abstract Text (1):
A method and apparatus for optimizing resource allocations is disclosed which utilizes the Karmarkar algorithm to proceed in the interior of the solution space polytope. The constraints on the allocation variables (the surfaces of the polytope) are partitioned into sparse and non-sparse partitions to permit applying a perturbation formula permitting rapid inversion of the resulting perturbed matrix products. Each successive approximation of the solution point, and the polytope, are normalized such that the solution point is at the center of the normalized polytope using a diagonal matrix of the current solution point, also partitioned into sparse and non-sparse portions. The objective function is then projected into the normalized space and the next step is taken in the interior of the polytope, in the direction of steepest-descent of the objective function gradient and of such a magnitude as to remain within the interior of the polytope. The process is repeated until the optimum solution is closely approximated.

Brief Summary Text (2):
This invention relates to systems for using the Karmarkar algorithm for resource allocation among a plurality of resource utilizers, and, more particularly, to the use of partitioning techniques for allocations including values heavily constrained by a large number of simultaneously operative constraints, thereby to improve the speed and efficiency of the Karmarkar algorithm.

Brief Summary Text (4):
The need for resource allocation decisions arises in a broad range of technological and industrial areas such as the assignment of transmission facilities in telephone transmission systems, the control of the product mix of a factory, the deployment of industrial equipment, inventory control, and others. Resource allocation in this context means, in general, the deployment of specific technological or industrial resources for the production of particular technological or industrial results.

Brief Summary Text (5):
Resource allocation decisions are typically subject to constraints on such allocations. Resources are always limited in overall availability, and, furthermore, the usefulness of a particular resource in some particular application may also be limited. For example, the traffic-carrying capacity of each individual link in a telecommunications system is limited, while the overall traffic offered to the communications system is also limited. Each particular allocation of resources can be associated with a "payoff," i.e., a cost of that allocation or an allocation benefit (e.g., profit). The problem, then, is to allocate the resources so as to satisfy all of the constraints and, simultaneously, to maximize the payoff, i.e., minimize the costs or maximize the benefits.

Brief Summary Text (6):
One method of representing such allocation decision problems is called the linear programming model. Such a model consists of a number of linear expressions that represent the quantitative relationships among allocations, constraints and payoffs. An expression is said to be linear if it is the sum of constant coefficients multiplied by unknown allocation values. Many resource allocation problems, of course, cannot be represented by such linear expressions, but involve higher powers of the unknowns or other nonlinearities in the relationships and hence are not susceptible to linear programming approaches.

Brief Summary Text (7):
It should be noted that the resource allocation problems discussed above are real physical problems arising in real physical systems. While it is true that significant quantitative aspects of the physical problem can be represented by the linear programming model, the purpose of this model is to provide optimum values which are then used in the physical world to construct or operate a physical system. Linear programming models are typically used to design telephone systems, to schedule airline activities or to control petro-chemical processes.

Brief Summary Text (9):
To find an optimum solution from among the many feasible solutions of a linear programming problem, an algorithm or procedure is applied. Such procedures follow an imaginary line or path from point to point in or on the surface of the polytope. The points on this path are steps in a series of interim solutions to the problem. Each such step is called an iteration. Each step or iteration, in turn, consists of the processing of equations that consider many interrelated constraints and variables.

Brief Summary Text (10):
It has long been known that the optimum solution to any linear programming problem must lie at one of the corners or vertices of the polytope. Successful algorithms or procedures for determining the optimum solution therefore follow a path which ultimately terminates at the optimum vertex. The speed of such algorithms depends, in part, on the number of steps and also, in part, on the complexity of each step.

Brief Summary Text (11):
One new method for solving linear programming models of physical allocation problems is called the Karmarkar algorithm, described in the article by N. K. Karmarkar entitled "A New Polynomial-Time Algorithm for Linear Programming," Combinatoria 4(4), pp. 373-395, 1984. Unlike the oler Simplex algorithm, which proceeds on the surface of the constraint polytope from vertex to vertex, the Karmarkar algorithm begins in the interior of the constraint polytope and proceeds in radial steps to the optimum vertex. At each iteration of the Karmarkar algorithm, the polytope space is rescaled to place the current value of the allocation at the polytope center. Because of the radial nature of successive steps, far fewer steps are required to reach the optimum vertex and hence the Karmarkar algorithm presents the possibility of being much faster than the Simplex algorithm, particularly for larger-sized problems.

Drawing Description Text (2):
FIG. 1 is a graphical representation of the Karmarkar method for determining optimum resource allocations in linear programming models;

Drawing Description Text (5):
FIG. 4 is a block diagram of a resource allocation system using the methods of FIG. 3 to control resource allocations.

Detailed Description Text (2):
The newly available Karmarkar algorithm for making optimum resource allocations with a linear programming model will first be discussed, and thereafter the modifications of the Karmarkar algorithm necessary to permit handling dense columns in the constraint matrix will be taken up.

Detailed Description Text (4):
In accordance with the Karmarkar algorithm, disclosed and claimed in the copending application of N. K. Karmarkar, Ser. No. 725,342, filed Apr. 19, 1985, now U.S. Pat. No. 4,744,028, and assigned to applicant's assignee, a starting point 51, located in the interior of polytope 50, is selected. As is well known, all points in the interior and on the surface of polytope 50 represent feasible solutions to the linear programming model. Also as is well known, in the absence of degeneracy, the optimum solution lies at one vertex of polytope 50, e.g., vertex 53. The Karmarkar algorithm proceeds radially entirely in the interior of polytope 50, in steps 52, 55, 56 . . . to successive points 54, et cetera, each closer to optimum point 53. Since the Karmarkar algorithm proceeds radially in the interior of polytope 50 instead of circumferentially on the surface, from vertex to vertex, as does the Simplex algorithm, the Karmarkar algorithm is inherently faster because it requires many fewer steps, particularly for larger models. The Karmarkar algorithm takes steps in the direction of the decreasing cost gradient in the polytope interior. Moreover, the Karmarkar algorithm involves rescaling the polytope space to equalize

or centralize the distance of each successive point in the trajectory or path from all of the facets of polytope 50, thereby normalizing the effect of the cost gradient in all of the dimensions of the polytope.

Detailed Description Text (50):
In FIG. 4 there is shown a process control system which controls a process 80. Process 80 may be a telephone communications system, a manufacturing process, a navigation process, or any other industrial or technological process which is to be optimized. A cost register 81 receives cost data on leads 82 representing the per unit costs of the various possible allocations of resources in controlled process 80. Cost data may be entered into register 81 from a computer terminal or from separate processes which dynamically determine these costs. While this cost data normally changes relatively slowly, there is nevertheless the ability to update this data via input leads 82 whenever necessary. If there are non-zero limits on the solution values, these limits, like the cost data, must be provided to LP controller 85 by way of a data input register like register 81.

Detailed Description Text (51):
Similarly, a limit register 83 is provided to store a representation of the total physical limits on each specific resource allocation. These limits are likewise relatively static and can be entered via leads 84 into register 83 from a computer terminal or from a separate limit-determining process. The outputs of registers 81 and 83 are applied to a linear programming (LP) controller 85 which carries out the process summarized in the flowchart of FIG. 3. LP controller 85 is, in the preferred embodiment, a programmed digital computer having stored therein the program which implements the flowchart of FIG. 3. Controller 85 may also comprise a complex of hardwired circuits designed to carry out the procedures of FIG. 3, a plurality of parallel processors to take advantage of the possibilities for parallel execution of the procedure, or a plurality of programmed linear arrays programmed for this purpose.

Detailed Description Text (55):
Depending on the complexity of the problem (the number of constraints sensed by sensors 86-88) and the stability of process 80, it is possible to more or less continually control process 80 by this method. Indeed, if the rate of change of the environmental factors sensed by sensors 86-88 is equal to or less than the rate of operation of LP controller 85, the process 80 will be controlled continuously. Higher rates of changes in the environment will introduce granularity into the control process, but will still permit near optimum operation, on the average, of the process 80. Indeed, given some history of the environmental changes, some predictive mechanism can be built into detectors 89-91 to predict the direction and magnitude of future changes in the outputs of sensors 86-88.

Detailed Description Text (56):
A typical type of problem in the telecommunications field to which the present invention can be applied is described in two articles in The Bell System Technical Journal, Vol. 60, No. 8, October 1981. A first article entitled "Design and Optimization of Networks with Dynamic Routing" by G. R. Ash et al. (p. 1787) describes the general telephone traffic routing problem while the second article, entitled "Servicing and Real-Time Control of Networks with Dynamic Routing," also by G. R. Ash et al. (p. 1821) describes an auxiliary problem of minimizing idle capacity due to erroneous predictions of traffic loads.

Detailed Description Text (57):
Other problems which would benefit from the new procedures herein described include industrial process control, deployment of personnel to provide customer services, blending of ingredients to form commercial products, oil refinery product mix, assignments of computer resources to a plurality of users, and many others. In each case, the cost (or benefit) coefficients must be measured or otherwise determined, the constraint limits must be established and the contributions of all of the decision variables to these constraints also measured or determined. The result of executing the procedures is, in each case, the specification of a set of control parameters which, when applied to the real world situation, will produce an optimum process or apparatus.

Detailed Description Text (59):
While the present inventor has constructed significant improvements on the Karmarkar method for solving linear programming models, it is to be understood that the claims of this invention relate only to the application of these novel improvements to

arrangements that determine the optimum allocation of resources in real world technological and industrial systems that lend themselves to a linear representation of the variables and constraints characterizing the system, i.e., physical arrangements that determine how resources are actually applied to optimize the performance of processes, machines, manufactures or compositions of matter. All other uses of the new method, such as computation research, algorithm research, or linear algebra research activities, form no part of the present invention. Similarly, use of the new method in non-technological or non-industrial systems likewise forms no part of the present invention. While the specification above employs mathematical notations to describe a new method for determining and adjusting values of system parameters, it is to be understood that the claims of this invention relate only to the method and apparatus that achieve this determination and adjustment. More specifically, the claims relate to a method and apparatus for determining and controlling the operational state of a commercial system so as to optimize the operational state of that system. In the context of this invention, a commercial system is a system that employs physical resources. Such a commercial system processes resources, incurs monetary (or equivalent) costs, and produces monetary (or equivalent) income. In such commercial systems, an optimized state of the system corresponds to a system state that optimizes a chosen attribute. Often, that chosen attribute is minimum attainable monetary costs incurred by the commercial system, of minimum attainable expenditure of the physical resources. All other uses of the method, such as mere computations to solve a given mathematical problem, computation research, algorithm research, or linear algebra research activities, are not encompassed by the claims appended hereto, and are not to be so construed. Similarly, the use of this new method on a non-technological or non-commercial systems likewise form no part of the present invention.

Other Reference Publication (6):
"On Projected Newton Barrier Methods for Linear Programming and an Equivalence to Karmarkar's Projective Method", Technical Report SOL 85-11, Systems Optimization Laboratory, Stanford University, Gill et al., Jul. 1985.

CLAIMS:

4. An optimized resource allocation system comprising:

a first plurality of physical resources available for use,

a second plurality of resource users using said physical resources; and

a controller for assigning said resource users to said physical resources so as to minimize the cost of providing said resources, said controller including

first means for iteratively and tentatively selecting feasible ones of said assignments in accordance with a preselected process until preselected stopping criteria are met,

second means for allocating said physical resources in accordance with a final one of said tentative assignment; and

third means for including the effect on successive selections of partitioning the users into a set of users that employ a number of said physical resources in excess of a preselected proportion of said physical resources, and a set of users that corresponds to the remaining ones of said users;

wherein

said first means of said controller constrains each of said feasible assignments to be represented within the interior of a normalized multidimensional convex feasible solution space.

5. A system for optimizing the performance of a controlled process in accordance with an optimizing criterion, said system comprising:

process control devices for controlling said process in response to variable control signal sets,

a plurality of sensors for sensing variable conditions affecting the operation of said process,

a plurality of data input devices for prescribing conditions affecting the operation of said process, and

a linear programming controller responsive to said sensors and said input devices for providing optimized control signal sets to said process control devices in accordance with the Karmarkar algorithm,

said controller including means for identifying successive tentative strictly feasible control signal sets including means for separating sparsely and non-sparsely constrained ones of said variable control signals, and means for selecting each of said successive tentative control signal set along the steepest gradient of a normalized version of said optimizing criteria.

10. A method for allocating industrial or technological resources, x.sub.j j-1,n, among a plurality of resource users subject to constraints ##EQU4## to optimize a cost function ##EQU5## said method comprising the steps of: tentaively allocating said available resources to said resource users in accordance with a specified deterministic process to form a tentative resource allocation characterized by a total cost that is lower than the cost of preceding tentative resource allocations;

repeating said step of tentatively allocating when said total cost fails to meet a preselected criterion; and

allocating said resource in accordance with the last of said tentative resource allocations when said total cost meets said preselected criterion;

wherein said step of tentatively allocating includes the steps:

(a) partitioning said constraints into sparse and non-sparse subsets S and N, respectively, of a constraint matrix, A=[a.sub.ij],

(b) selecting an initial allocation x meeting said constaints,

(c) determining a dual variable w in accordance with the formula

w={(SD.sub.S.sup.2 S.sup.T).sup.-1 -(SD.sub.S.sup.2 S.sup.T).sup.-1 N(D.sub.N.sup.-2

+N.sup.T (SD.sub.S.sup.2 S.sup.T).sup.- N).sup.-1 N.sup.T (SD.sub.S.sup.2 S.sup.T).sup.-1 }AD.sub.x.sup.2 c,

where S is the submatrix of A corresponding to sparse columns of A, where a column j.sub.o is sparse when the number of nonzero elements in column j.sub.o is less than a preselected proportion of the total number of elements a.sub.ij.sbsb.o for i=1, 2, . . . , m, N is the submatrix of A corresponding to the non-sparse columns, D.sub.x is a diagonal matrix containing the elements of the current allocation vector x D.sub.S is a diagonal matrix containing those elements of the current allocation vector x that correspond to the sparse columns, and D.sub.N is a diagonal matrix containing those elements of the current allocation vector x that correspond to the non-sparse columns,

(d) determining the direction of the next iteration of an iterative procedure for approximating said optimal cost in accordance with the formula

z=D.sub.x.sup.2 (c-A.sup.T w),

where D.sub.x is the diagonal matrix of the current allocation values given by ##EQU6## (e) determining a complementary slackness coefficient .gamma. of said current iteration from the formula

.gamma.=max.sub.j z.sub.j /x.sub.j, and

(f) determining a dual feasibility coefficient .delta. of said current iteration from the formula

.delta.=-min.sub.j z.sub.j /x.sub.j.sup.2,

wherein said step of returning to said step of tentatively allocating allocating

includes the steps:

(g) testing said current iteration against the inequality

.gamma.+M.delta.<(.epsilon./n)(.vertline.c.multidot.x.vertline.+1),

where

M=max.sub.j x.sub.j

and .epsilon. is an arbitrary small error measure,

(h) if the test of step (g) is not satisfied, returning to step (c) with the new allocations

x.rarw.x-.alpha.z/.gamma.,

where

.alpha. is an arbitrary real number strictly between zero and one, and

wherein said step of allocating allocates said resources in accordance with said current allocation when the test of step (g) is satisfied.

WEST

Generate Collection Print

L14: Entry 18 of 22 File: USPT Nov 12, 1996

DOCUMENT-IDENTIFIER: US 5574911 A
TITLE: Multimedia group resource allocation using an internal graph

Abstract Text (1):
An intelligent system for the efficient selection and allocation of the various types of resources available in a multimedia environment. The system interrelates a combination of user input parameters with the resident hardware and software parameters of the requesting multimedia resources by grouping into common computing needs. In addition to traditionally known hardware/software parameters the system recognizes specific limitations of resources which would effect a proper multimedia presentation to the end user. The system uses an internal graph structure to interrelate the various resources as they are allocated to proved for an intelligent data flow.

Brief Summary Text (3):
The present invention relates to an apparatus and method for the allocation of multimedia system resources. More, particularly, the present invention relates to an intelligent mechanism for group allocation of multimedia system resources based on device access requests. The invention incorporates the particular limitations of the end user as well as the constraints and priorities of the requesting multimedia devices.

Brief Summary Text (8):
Some prior art systems have attempted to make piece-meal resource allocations. They allow each device to request resources, but since each request is separate, resources allocation occurs with only mixed benefits to the end user. Other prior art systems have made specially coded objects which are tightly coupled in order to make better commitments to resources. This attempt at resource allocation severely limits future development and expandability.

Brief Summary Text (9):
The apparent standard prior art answer to resource allocation is to use existing operating system mechanisms to solve the problem. Most operating systems, for instance, will allow system resources (i.e. memory) to be allocated to a single device/user on a first come basis. This typical prior art mechanism has been common for over 20 years. This method, referred to as a default type allocation, cannot efficiently and properly interrelate a multiplicity of simultaneous or copending multimedia devices.

Brief Summary Text (10):
One step better than the default mechanism has been implementing a special purpose dedicated CPU scheduler or a special network bandwidth allocator to manage resources. This particular solution allows limited intelligence to be integrated into resource allocation, but fails to account for user and device specific constraints.

Brief Summary Text (11):
The following patents are examples of prior art system resource allocation methods. While the references show resource allocation methods, they fail to account for the hardware, software and user constraints as required in a multimedia application versus the discrete data requirements in a non-multimedia application.

Brief Summary Text (13):
U.S. Pat. No. 4,937,743 to Rassman, et al. covers the scheduling of resources over time. These resources are allocated to one owner at a time, and provisions are made for automatic rescheduling if conflicts occur. This kind of resource management

differs from the one presented herein, and does not address the issue of time, but rather addresses partial ownership and division of resources at any given point in time. The patent implies (through its use of database) that access and decision making is distributed. That is, that each scheduling application makes the decisions necessary and the only central information is the database itself.

Brief Summary Text (14):
U.S. Pat. No. 5,208,745 to Quentin, et al. concerns the use of an interface, scripting language and. expert system to solve end-user problems. No mention is made of resource allocation and.backslash.or management thereof.

Brief Summary Text (15):
U.S. Pat. No. 5,025,395 to Nose, et al. addresses the idea of objects and the need for objects to have resources associated with them. Its focus is completely on the presentation of these objects in a user interface and the method by which an end-user associates resources with an object. The applications described in the patent allow the user to select a group of objects, but these applications rely on the default or existing system mechanisms for the allocation of these resources, and they do not specify any collaboration or grouping of the objects internal to the system.

Brief Summary Text (17):
As illustrated above, the prior art has failed to fully integrate a multiplicity of multimedia requests. Therefore, it is an object of this invention to accurately, efficiently and in a timely manner, allocate multimedia computing resources to a plurality of multimedia devices or groups of devices which are requesting access to the system resonrces.

Brief Summary Text (18):
Another object of the invention is to provide for a centralized entity and a heuristic procedure by which computer resources can be requested and allocated.

Brief Summary Text (20):
It is a further object of the invention to provide for a graph type data flow structure to properly allocate system resources.

Brief Summary Text (23):
The present invention specifies a technique for allocating system computing resources to groups of requesting multimedia objects or processes. By collecting all knowledge about the constraints of a particular data processing element, of a group of elements or those as imposed by the user, effective allocation of computing resources can take place.

Detailed Description Text (4):
In order to get optimal multimedia computing resource allocation the processing element (object) requests are to be grouped by type. A good example of a case where grouping is useful is a teleconference as shown in FIG. 1. This figure shows two computers on a network sending video and audio back and forth. The two graphs are obviously disconnected, but should be considered as grouped because of the need for allocation of common system resources for each graph. The allocation of computing resources to groups of multimedia processes and objects often depends on system and user constraints and/or limitations. If these constraints are ignored, poor resource allocation will ultimately occur. The system of the instant invention collects the particular constraint information germane to each device, groups of devices and/or applications together to form a centralized management entity, which for this application will be referred to as a "Resource Manager".

Detailed Description Text (6):
The procedure for collection of constraints starts when the user passively places constraints on the resource allocation by selection of particular devices or programs. The end user is meant to be as unaware as possible about resource needs. The user makes decisions about resource constraints by selecting a given application to run. For instance, if the user chooses to play a movie from a server, one set of constraints is implied. If the user had chosen to video-conference with someone other constraints would have been implied. In some cases, the user has other controls such as a video control panel where the user can request what quality of video to use (for example, number of frames per second and frame size).

Detailed Description Text (8):

The actual multimedia objects/processes are then responsible for using the application, its constraints on performance and user given constraints to calculate details about actual resource usage. For playing a movie, a lot of memory might be required to do buffering and remove jitter. For doing video conferencing, only small buffers would be required, but getting CPU time may be very important.

Detailed Description Text (9):
Applications are expected to know about priorities and be able to place them on requests of all types. Any request to allocate resources can request priorities be given to certain constraints. A common example to this is that, besides specifying the end-to-end delay bounds, the application can request that the delay be considered the most important factor in resource allocation. Hence, priorities can be placed on resource types by the application. In addition, control panel applications may be written which expose to the user details of various objects. Users can to control network priorities or assign relative priorities to applications. With such a control panel the user may set an alarm buzzer to have a higher priority than a mood music player, and the system would account for this.

Detailed Description Text (10):
Requests for ways to optimize resource use can be included. Each object reports the bounds on resource needs that are acceptable for its proper execution. Each object then describes any special parameters (control values) which can be modified to improve or degrade resource needs or data quality. This description also informs the Resource Manager as to what resources are affected by each parameter.

Detailed Description Text (16):
To allocate resources without graph knowledge, the objects would have to talk among themselves and come to a consensus within some external resource allocator. There would be no easy way to determine when all the objects had reported in, so if for any reason (say one of the objects was across a network link) an object reported in late, the system would have to wait an indefinite period of time for completion. In addition, by giving the knowledge of connectivity to another entity, objects are freed from the need to know how they are connected. This leads to other benefits when implementing the solution (such as simplified interfaces).

Detailed Description Text (29):
Step 8: At this point, the Graph Manager knows all about the Graph. The application can now do any operations on the graph that it considers valuable. In the present, the primary operation is resource allocation. This is done with "Prepare Device" function.

Detailed Description Text (34):
Once all the requests are collected, as determined by the Graph Manager, the Resource Manager balances the various constraints on resource allocation and attempts to come up with a successful match. When the number of constraints is known in advance, the algorithm for allocation is determinable in advance. When allowing any number of resources (N) and any number of parameters (M) to affect the solution, a more generic algorithm must be used. The instant invention proposes an iterative, heuristic approach to solving the allocation problem.

Detailed Description Text (35):
As discussed above, if all the possible resources were known in advance, it might be possible to generate a solution space or algorithm for resource allocation in advance. Many systems for handling specific resource constraint sets have been proposed, but general purpose solutions cannot rely on the knowledge of the resource constraints at the time of the writing of the resource management code. In the general case, the Resource Manager needs to handle some unknown number of resources with an unknown number of constraints.

Detailed Description Text (36):
If the constraints can be met with available resources on the system, the Resource Manager can allocate the resources to the various objects at that time.

Detailed Description Text (37):
If one or more of the constraints asks for a computing resource to be allocated at greater than what is currently available for allocation, or if the constraints are to limiting with regards to a resource, one of the parameters that affects that resource is modified and another check is made against the constraints.

Detailed Description Text (38):
It is likely that lowering the use of one kind of resource may increase the use of another kind, and that this process of modifying parameters and rechecking resources is repeated a number times before a solution is found or before it is discovered that not solution can be found. If no solution is found, then the resource allocation request is rejected.

Detailed Description Text (40):
The algorithm described in this invention approaches resource allocation as a problem in which `N` resource types are available for objects to share. This set of resources is unlikely to change between attempts to allocate resources. Each resource type has an allocation algorithm associated with it. These algorithms are encapsulated into objects called "Resource Controllers". It is the job of a Resource Controller to dole out the resource it controls. A Resource Controller could control a computing resource (such as CPU time or memory space) or a physical resource (such as a disk drive or audio playback adapter). This object is expected to implement a useful algorithm for reservation and allocation. There are many instances of prior art for such algorithms, and this disclosure does not attempt to cover the subject of such algorithms. The important statement is that there are `N` resources, each with some object managing its allocation.

Detailed Description Text (41):
For any given attempt at allocation, a group of one or more objects may be interested in resources. Each of these objects may place constraints on their resource needs. These constraints are usually specified in terms of bounds. For example, an object could say that it desires the use of 2.0 megabytes of RAM and that it will accept a minimum of 1.3 megabytes of RAM. Thus, bounds are placed on the allocation of various resources, and it is up to the Resource Manager to take a global view of all the requests and find a solution where all the objects meet their desired value if possible, but at least meet their minimum value.

Detailed Description Text (44):
The preferred embodiment of the instant invention uses parameters called Control Values (as previously mentioned). A Control Value is an alterable parameter on a object which will cause it to modify its resource needs. Each object may report as many Control Values as it wants to the Resource Manager. A Control Value is defined by specifying the desired value for the parameter, a worst-case value for the parameter, a priority, and a list of what resource types it will affect. Thus, the Resource Manager is given information as to what "knobs" on the object can be turned, and when one is turned, what resource type's usage will decrease.

Detailed Description Text (47):
The preferred embodiment uses a heuristic algorithm. During each allocation phase, the Resource Manager collects the information (as above) and then attempts to make an allocation using the desired values for each resource (via calls to the Resource Controller). If this succeeds, then the current allocation is completed. If, however, there was not enough of one or more resources, then the Resource Manager looks at the list of failed resource types, and compares it to the list of Control Values. Since each Control Value specifies what resources it can decrease the resource needs of, this eliminates many of the Control Values from determination in solving the problem. Once the smaller list of Control Values that actually affects the failed resource types is determined, the lowest priority Control Value is lowered from its desired value toward its worst value. Then, resource requests are collected again and another attempt at allocation is made. This kind of process is iterated on until either successful solution is reached, or all the Control Values affecting a scarce resource are at their minimum values and still no solution has been found.

Detailed Description Text (51):
As shown in FIG. 6, the various processing elements in the system interact in order to complete the resource allocation phase.

Detailed Description Text (55):
3. Application requests that resources be allocated. This results in a call to the Graph Manager to do reservation, then allocation of resources.

Detailed Description Text (62):
7. The Graph Manager calls the "Resource Manager" to balance the resources. If failure occurs, the "Resource Manager" returns the list of resource types that

failed. The "Graph Manager" then modifies the appropriate Control Value(s). If no Control Values are appropriate to the resource type, then the resource allocation returns failure. If at least one could be modified, then return to step 6 and try again. Reservation of resources actually marks them as being "in use" but does not allocate them.

Detailed Description Text (63):
8. Once a successful reservation has been made, the Graph Manager calls each object (in a traversal like that of step 6) to let them confirm their acceptance of the granted resources. If any object disagrees with what it is granted, then resource allocation fails.

Detailed Description Text (64):
9. The Graph Manager calls each object to allocate resources (in a traversal like that of step 6). Each object uses the a Get Resources function of the Resource Manager to actually allocate the resources and return information about the resources to the object. The Resource Manager uses the Resource Controllers to complete this task.

Detailed Description Text (65):
10. Resource allocation is now complete, so the Graph Manager calls the Resource Manager to free the FlowID.

Detailed Description Text (70):
A system and method has been shown in the above embodiments for the effective grouping and allocation of computing resources to a plurality of requesting multimedia objects or elements. While various preferred embodiments have been shown and described, it will be understood that there is no intent to limit the invention by such disclosure, but rather, is intended to cover all modifications and alternate constructions falling within the spirit and scope of the invention as defined in the appended claims.

CLAIMS:

1. In a computer-based multimedia system, a computing resource allocation system having dynamic allocation of requests by a plurality of objects comprising:

means for creating said plurality of objects and connection thereof;.

means for grouping said requests;

means for collecting hardware, software or flow limitations and constraints associated with each of said plurality of objects and associated connections;

means for determining the allocation of said computing resources based on said grouping and said collected limitations and constraints:

internal graph means to coordinate said allocation of said requests for said computing resources with each of said computing resources responding to said internal graph means, and

means responsive to said internal graph means to allocate said computing resources to said requesting objects.

2. A computing resource allocation system as in claim 1, wherein if sufficient resources are available to satisfy the requests of a given group, the means to allocate allocates the requested resources to each requester.

3. A computing resource allocation system as in claim 1, wherein if sufficient resources to satisfy said requests of a given group are not available, the means for determining determines a scenario which enables the resource allocation system to comply with all of said requests while remaining within the bounds of each requestors hardware/software constraints.

4. A computing resource allocation system as per claim 1, wherein said means for determining is implemented with a heuristic algorithm.

5. A computing resource allocation system as per claim 1, wherein said internal graph means comprises a plurality of nodes and edges.

6. A computing resource allocation system as per claim 1, wherein said internal graph means is collapsed and stored as arrays or linked structures.

7. A method of allocation of multimedia computing resources to a plurality of requests from a plurality of processing elements comprising:

collecting hardware, software or flow constraints associated with said requesting elements;

collecting variable parameters associated with each of said requesting elements;

collecting user specific constraints associated with selected applications to be run during said allocation of said multimedia computing resources;

grouping all of said requests by type;

determining the allocation of said multimedia computing resources based on said hardware and software constraints, said variable parameters, said user specific constraints and said request types;

creating an internal graph structure representing said allocation;

traversing said internal graph to coordinate the proper sequence of said requesting elements, and contacting each of said requesting elements to confirm acceptance of said sequence and, upon acceptance, allocating said computing resources to said requesting objects.

8. The method of resource allocation as per claim 7, wherein said determining step is implemented with a heuristic algorithm.

9. The method of resource allocation as per claim 7, wherein said variable parameters represent a range of acceptable operation.

10. The method of resource allocation as per claim 9, wherein said method provides for flexible allocation and deallocation of said resources based on said range of acceptable operation.

11. The method of resource allocation as per claim 10, wherein said constraints of each requesting device are dynamically modified based on said flexible allocation and deallocation.

12. The method of resource allocation as per claim 7, wherein said internal graph structure comprises a plurality of nodes and edges.

13. The method of resource allocation as per claim 7, wherein said internal graph structure is collapsed and the collapsed graph is stored as arrays or linked structures.

14. In a computer-based multimedia system, a computing resources allocation method having dynamic allocation of requests by a plurality of objects comprising the following steps:

selection of a computer application;

said application creating a set of objects and connecting them;

said application requesting allocation of said computing resources from a resource manager;

said graph manager collects flow identification from said resource manager;

said graph manager collects control values from said objects;

said graph manager interacts with said resource manager to optimize balancing of said computing resources based on said flow identification and control values;

said graph manager requests acceptance of said optimized balancing from said objects, and

upon acceptance, said graph manager calls each object to allocate resources.

15. In a computer-based multimedia system, a computing resources allocation method having dynamic allocation of requests by a plurality of objects, as per claim 14, wherein said optimize balancing step is implemented with an artificial intelligence method.

16. In a computer-based multimedia system, a computing resources allocation method having dynamic allocation of requests by a plurality of objects, as per claim 14, wherein said optimize balancing step is implemented with a heuristic algorithm.

WEST

Generate Collection Print

L14: Entry 15 of 22 File: USPT Nov 16, 1999

DOCUMENT-IDENTIFIER: US 5987021 A
TITLE: Method and apparatus for allocating resources between queued and non-queued services

Abstract Text (1):
In a resource allocator (106), a reserved resource group comprising at least one communication resource is maintained for use in supporting non-queued services. When a request for the at least one non-queued service is received, a communication resource from the reserved resource group is allocated to the request (305). When the reserved resource group becomes depleted, at least one more communication resource is assigned to the reserved resource group with greater preference relative to allocation of communication resources to requests for at least one queued service (307). In a preferred embodiment, a communication resource is assigned to the reserved resource group prior to allocating resources to any queued service requests. Additionally, communication resources (200) having varying grades of service can be accommodated.

Brief Summary Text (2):
The present invention relates generally to wireless communication systems and, in particular, to a method and apparatus for allocating resources between queued and non-queued services provided therein.

Brief Summary Text (6):
Still another approach to this problem is to partition the system resources in a predetermined manner so that no one service can dominate the resources at the expense of the other services. In one approach referred to as "hard" partitioning, the resources are partitioned in accordance with historical usage of the respective services such that requests for each type of service can only be fulfilled from the designated portion of resources. For example, if, in a given system, 70% of system capacity is historically used to service dispatch requests, with the remaining 30% used to service telephone requests, the system resources will likewise be partitioned on a 70/30 basis. This works well so long as the actual system load stays at 70% dispatch and 30% telephone. However, should the amount of dispatch requests decrease, the otherwise unused resources assigned to support dispatch cannot be used to support an increase in telephone services. Conversely, should the amount of dispatch requests increase, the system cannot assign additional resources to support the increased requests. These inefficiencies likewise apply to increases and decreases in telephone requests.

Brief Summary Text (7):
A somewhat "softer" variant of hard partitioning is to always maintain a minimum number of resources for a given service. For example, at system set-up, the resource allocator in the system would be configured to ensure that no less than N resources (where, typically, N>1) are available and/or in use to support a given service. As a result, the given service will always be guaranteed a minimum level of resources for use in supporting requests for that service, thereby ensuring at least a minimum level of service at all times. Should the number of requests for that service increase such that the N resources are not enough to support all requests immediately, the system is free to assign additional resources, if available, to service the additional requests. However, it is still possible that there are more than enough reserved idle resources to meet the demand for that service at any given time. As a result, resources that may be put to better use in support of other services remain unused in order to maintain the required minimum number for the given service. In effect, both the hard partitioning and minimum number methods work well so long as the offered system load at any given time matches the historical load upon which the partitioning/minimum thresholds are based, but both introduce

inefficiencies when the offered load varies from the historical basis.

Brief Summary Text (8):
The problems described above are further exacerbated when the communication resources themselves provide varying grades of service and where different services require communication resources having minimum grades of service. This is particularly true where requests for non-queued services must be fulfilled using resources having a higher grade of service and where requests for queued services may be fulfilled using either lower or higher grade resources. Therefore a need exists to better accommodate the ability of non-queued services to obtain requested resources in a multi-service shared infrastructure communications system.

Drawing Description Text (4):
FIG. 3 is a flowchart of a method for use in a resource allocator in accordance with a preferred embodiment of the present invention.

Detailed Description Text (2):
Generally, the present invention provides a method and apparatus for allocating resources between at least one queued service and at least one non-queued service. In particular, a reserved resource group comprising at least one communication resource is maintained for use in supporting the non-queued services. In effect, the reserved resource group does not rely upon historical usage patterns, but rather tracks the current system load distribution at all times. When a request for the at least one non-queued service is received, a communication resource from the reserved resource group is allocated to the request. If the reserved resource group becomes depleted as a result, at least one more communication resource is assigned to the reserved resource group with greater preference relative to allocation of communication resources to requests for the at least one queued service. In a preferred embodiment, the greater preference is effectuated by assigning a communication resource to the reserved resource group prior to allocating resources to any queued service requests. Furthermore, the method can accommodate communication resources having varying grades of service. In a preferred embodiment, the method is carried out by a resource allocator. In this manner, the present invention ensures maximum resource usage efficiency while maintaining perceived system quality, particularly system availability.

Detailed Description Text (3):
The present invention may be more fully described with reference to FIGS. 1-3 and accompanying text. FIG. 1 illustrates a wireless communication system 100 comprising a resource allocator (access control gateway or ACG) 106 configured in accordance with the present invention and coupled to at least one non-queued service processor 102 supporting at least one non-queued service and at least one queued service processor 104 supporting at least one queued service. Communications systems supporting multiple services are known in the art, an example of which is described in U.S. Pat. No. 5,548,631 entitled METHOD AND APPARATUS FOR SUPPORTING AT LEAST TWO COMMUNICATION SERVICES IN A COMMUNICATION SYSTEM, issued Aug. 20, 1996 to Krebs et al. and assigned to Motorola, Inc., which patent is incorporated herein by this reference.

Detailed Description Text (4):
The at least one queued service may comprise, but is not limited to, dispatch services, as known in the art. Likewise, the at least one non-queued service may comprise, but in not limited to, telephone service, control service and mobility service, as known in the art. In order to provide the queued and non-queued services to subscriber units 122-124, the resource allocator 106 is coupled to a plurality of radio base stations 110-114 which transceive a plurality of communication resources 116-120 such as RF carriers supporting frequency-division multiplexed (FDM), time-division multiplexed (TDM) and/or code-division multiplexed (CDM) protocols. The resource allocator 106 includes a processor 107 (such as a computer, microprocessor, microcontroller, digital signal processor or combination thereof as known in the art) coupled to memory 108 (such as volatile and non-volatile memory devices, as known in the art) suitable for storing software programming instructions and/or operating parameters.

Detailed Description Text (5):
In this particular embodiment, the resource allocator 106 determines the radio services requested by the subscriber units 122-124, via the base stations 110-114 and relays the request to the appropriate processor 102-104. To support the request for radio service, the resource allocator 106 receives service approval messages

from the processors 102-104 and relays this information to the subscriber units 122-124 via the base stations 110-114. Additionally, the resource allocator 106 issues communication resource assignments to the subscriber units 122-124 via the base stations 110-114. Operation of the resource allocator 106 is described in further detail below with reference to FIG. 3.

Detailed Description Text (8):
Referring now to FIG. 3, a flowchart of a method for use in a resource allocator in accordance with a preferred embodiment of the present invention is depicted. In the preferred embodiment, the steps illustrated in FIG. 3 and described below are implemented as software routines executed by a resource allocator 106, as known in the art. Alternatively, it is recognized that implementation of the method may be distributed amongst various platforms depending on the architecture of the communication system. Regardless, the method begins at step 301 where at least one communication resource from a plurality of communication resources is assigned to a reserved resource group used to support non-queued services. The communication resources may take a variety of forms such as periodically repeating time slots (as in the preferred embodiment), distinct carrier frequencies, orthogonal codes, etc.

Detailed Description Text (9):
The present invention draws a distinction between resources that are assigned and resources that are allocated. In particular, an allocated resource is a resource that has been designated for use in an ongoing communication and is therefore currently unavailable to support other communications regardless of the type of service. In contrast, an assigned resource is a resource that has been designated for use in a future communication of a given type, and is therefore currently unavailable to support types of services other than the given type. Thus, the reserved resource group, after step 301, is essentially a list of one or more communication resources reserved for later use in support of non-queued services. Until allocated, the communication resources within the reserved resources group remain idle. Furthermore, in the preferred embodiment, the at least one communication resource assigned at step 301 provides the first grade of service, as described above.

Detailed Description Text (10):
Steps 302 and 303 illustrate the preferred method for handling the deallocation of communication resources, i.e., resources that become available as ongoing communications are concluded. When, at step 302, it is determined that one or more resources have become deallocated, the deallocated resources are released back into the plurality of communication resource at step 303, as opposed to being immediately assigned to the reserved resource group. This serves the purpose of allowing the resource allocator the chance to "churn" the communication resources being assigned to the reserved resource group, thereby continually optimizing channel configuration.

Detailed Description Text (11):
Assuming that a resource has not been deallocated, the process continues at step 304 where it is determined whether a request for service, either queued or non-queued, has been received. Of course, if no request for any service has been received, there is no need to allocate any resources and the process can continue at step 302. However, when a request for a non-queued service is received at step 304, the resource allocator (after having received the necessary service approval messages from the service processor supporting the requested non-queued service) allocates a communication resource from the reserved resource group in support of the requested non-queued service at step 305. In effect, this means that the allocated resource is removed from the list designating the reserved resource group.

Detailed Description Text (12):
At step 306, it is determined whether the reserved resource group has been depleted. As shown in FIG. 3, this determination is made whenever a resource has been deallocated or after a resource has been allocated from the reserved resource group. In the context of the present invention, the reserved resource group is "depleted" when it is less than full. When the reserved resource group comprises only a single resource at any time, the allocation of that single resource will obviously deplete the reserved resource group. If the reserved resource group is not depleted, processing continues at step 302.

Detailed Description Text (13):
However, if the reserved resource group does become depleted, at least one other

communication resource of the plurality of communication resources is assigned to the reserved resource group with a greater preference relative to allocation of resources to queued service requests at step 307.

Detailed Description Text (14):
In the preferred embodiment, this "greater preference" is achieved by assigning the at least one other resource before allocating any resource to queued service requests. Where resources having first and second grades of service are available, a further refinement is possible in that only a resource having the first grade of service is assigned to the reserved resource group prior to allocating a resource having the second grade of service to a queued service request.

Detailed Description Text (15):
It is recognized that other criteria for assigning the at least one other resource could be used. For example, assignment to the reserved resource group could take priority only over allocation of resources to non-priority (e.g., non-emergency) queued service requests. Alternatively, only users of non-queued services that have paid an extra premium for an overall higher grade of service may be given preference. Other examples are no doubt easily conceived by those having ordinary skill in the art. At a minimum, the criteria used must ensure that non-queued service requests will not be starved of resources by queued service requests.

Detailed Description Text (16):
In order to assign the at least one other resource to the reserved resource group, the resource allocator may select a currently unused communication resource from the plurality of communication resources. However, an unused resource may not always be available at any given moment. In the preferred embodiment, this occurrence causes queued requests to be blocked while waiting for deallocation of resources. Alternatively, it is anticipated that the resource allocator could pre-empt usage of communication resource(s) in one or more currently ongoing communications. For example, low-priority calls could be terminated and the now-available resources used to fulfill the necessary assignment to the reserved resource group.

Detailed Description Text (17):
In yet another alternative, an ongoing communication could have its resource re-allocated to it such that no interruption in service is experienced, but such that a communication resource that was otherwise unavailable is now free. This is possible, in the preferred embodiment, by converting a high grade communication resource (e.g., an i3 resource) into two lower grade resources (e.g., two i6 resources), with one lower grade resource re-allocated to the ongoing communication, and the other lower grade resource now available for assignment. Additionally, it may be possible to re-arrange the order of the time slots used to provide the communication resources such that an additional higher grade resource is realized from previously idle lower grade resources.

Detailed Description Text (18):
By assigning resources to the reserved resource group with greater preference relative to allocation to queued services, the present invention effectively allows the resource allocator to keep just ahead, but not too far ahead, of the need for resources to support the non-queued services. Rather than pre-reserving a plurality of resources that may not be enough, or that may be more than enough, to service the current demand for non-queued services, the present invention effectively tracks the current system loading and provides just enough reserved resources to support the non-queued services.

Detailed Description Text (19):
If, at step 304, a request for a queued service is received, the process proceeds to step 308 where it is first determined whether the reserved resource group is depleted. In the preferred embodiment, requests for queued services that only use resources having the second grade of service are allocated resources only when the reserved resource group is not depleted. Thus, if the reserved resource group is not depleted, a communication resource is allocated to the request for the queued service at step 309. The communication resource allocated at step 309 may be obtained from the plurality of communication resources, i.e., those resources currently unused and not assigned to the reserved resource group. Alternatively, as discussed in the example below, the allocated resource may be obtained from a "holdback" (analogous to the reserved resource group discussed above) created specifically for that purpose. If, however, the reserved resource group is depleted, the request is refused at step 310, after which the request remains queued.

Detailed Description Text (21):
In the simulation, three reserved resource groups (or holdbacks) were created, called "i3hold", "i6hold" and "i12hold". The minimum number of resources to be maintained in each holdback is governed by a minimum threshold value for each, respectively labeled "min3", "min6" and "min12". Thus, on startup, the pseudo-code illustrated in Table 1 is carried out.

Detailed Description Text (23):
As shown in Table 2, if the i3 holdback is disabled (min3=0), then get the resource directly from the plurality of resources. Otherwise, get the resource from the i3 holdback. Regardless, once the i3 resource has been allocated, go to the "MaintainHoldbacks" function which, as described below, serves to restore the holdbacks, if necessary. The process for allocating i12 resources for control services is similar except, of course, that the i12 holdback is used.

Detailed Description Text (25):
Thus, when the i6 holdback is disabled (min6=0), the resource allocator will get the resource from the plurality of resources only if the i3 holdback is not depleted. If the i3 holdback is depleted, the i6 request is denied. Alternatively, if the i6 holdback is not disabled, the i6 resource is taken from the i6 holdback. In either case, MaintainHoldbacks is called to replenish the respective holdbacks as needed. By denying the i6 request when the i3 holdback is depleted, the resource allocator allows an i3 resource to become available and assigned to the i3 holdback.

Detailed Description Text (27):
Once again, the request for the i6 resource will be processed only if the i3 holdback has not been depleted. Since dispatch is a queued service, it is not necessary to resort to a holdback to get a resource; if a resource is not currently available from the plurality of resources, the request will remain queued. Note also that MaintainHoldbacks does not need to be called after allocating the i6 resource for dispatch since none of the holdbacks have been affected.

Detailed Description Text (30):
In the simulation, a cell comprising three radio frequencies, each supporting time slots as described above relative to FIG. 2, was modeled. A 2% blocking percentage for phone and 5% blocking percentage for dispatch was assumed. The three resource allocation techniques discussed herein were tested: hard partitioning, minimum threshold partitioning, and the present invention. Table 6 illustrates the parameters used in testing each method.

Detailed Description Text (31):
As shown in Table 6, various loading mixes of i3 telephone and i6 dispatch were used. For each loading mix, the operating parameters were manually adjusted to optimize performance for the hard partition and minimum threshold methods; the parameters for the present invention were not altered to track any particular load mix. In particular, the i12 and i3 holdbacks were set to allow for one resource to be assigned to each, whereas the i6 holdback was disabled.

Other Reference Publication (1):
Algorithm to Provide Dynamic Channel Allocation of 3:1 Resources By Patrick J. Keegan, Matthew W. Simpson and Gary J. Goethals, Motorola Technical Developments, vol. 32, Sep. 1997.

CLAIMS:

1. In a wireless communication system comprising a plurality of communication resources, the wireless communication system supporting a plurality of services that include at least one queued service and at least one non-queued service, a method for allocating resources between the at least one queued service and the at least one non-queued service, the method comprising steps of:

assigning at least one communication resource of the plurality of communication resources to a reserved resource group used for servicing the at least one non-queued service;

allocating a communication resource from the reserved resource group to the at least one non-queued service; and

when the reserved resource group is depleted, assigning at least one other communication resource of the plurality of communication resources to the reserved resource group with greater preference relative to allocation of the plurality of communication resources to the at least one queued service.

6. The method of claim 5, further comprising a step of:

allocating a communication resource providing the second grade of service to the at least one queued service only when the reserved resource group is not depleted.

7. The method of claim 1, wherein the step of assigning the at least one other communication resource further comprises assigning the at least one other communication resource prior to allocating any of the plurality of communication resources to the at least one queued service.

8. The method of claim 1, the step of assigning the at least one other communication resource further comprising a step of:

providing the at least one other communication resource by performing at least one of: pre-empting usage of communication resources in an ongoing communication, re-allocating communication resources to the ongoing communication, and blocking one of the plurality of services.

9. The method of claim 1, further comprising steps of:

when any of the plurality of communication resources are deallocated to produce a deallocated communication resource and when the reserved resource group is depleted, releasing the deallocated resource to the plurality of communication resources; and

assigning yet another communication resource of the plurality of communication resources to the reserved resource group with greater preference relative to allocation of the plurality of communication resources to the at least one queued service.

10. In a wireless time division multiple access (TDMA) communication system comprising a plurality of periodically repeating time slots forming a plurality of communication resources, the wireless TDMA communication system supporting a plurality of services that include at least one queued service and at least one non-queued service, a method for allocating resources between the at least one queued service and the at least one non-queued service, the method comprising steps of:

assigning a first communication resource of the plurality of communication resources and providing a first grade of service to a reserved resource group used for servicing the at least one non-queued service;

allocating a communication resource from the reserved resource group to the at least one non-queued service; and

when the reserved resource group is depleted, assigning at least one other communication resource of the plurality of communication resources and providing the first grade of service to the reserved resource group prior to allocating a second communication resource of the plurality of communication resources to the at least one queued service, wherein the second communication resource provides a second grade of service inferior to the first grade of service.

14. The method of claim 10, further comprising a step of:

allocating the second communication resource to the at least one queued service only when the reserved resource group is not depleted.

15. The method of claim 10, the step of assigning the at least one other communication resource further comprising a step of:

providing the at least one other communication resource by performing at least one of: pre-empting usage of communication resources in an ongoing communication, re-allocating communication resources to the ongoing communication, and blocking one of the plurality of services.

16. The method of claim 10, further comprising steps of:

when any of the plurality of communication resources are deallocated to produce a deallocated communication resource and when the reserved resource group is depleted, releasing the deallocated resource to the plurality of communication resources; and

assigning yet another communication resource of the plurality of communication resources and providing the first grade of service to the reserved resource group prior to allocating the second communication resource to the at least one queued service.

17. A resource allocator for use in a wireless communication system comprising a plurality of communication resources, the wireless communication system supporting a plurality of services that include at least one queued service and at least one non-queued service, the resource allocator comprising:

means for assigning at least one communication resource of the plurality of communication resources to a reserved resource group used for servicing the at least one non-queued service;

means for allocating a communication resource from the reserved resource group to the at least one non-queued service; and

means for assigning, when the reserved resource group is depleted, at least one other communication resource of the plurality of communication resources to the reserved resource group with greater preference relative to allocation of the plurality of communication resources to the at least one queued service.

18. The resource allocator of claim 17, wherein the plurality of communication resources comprises communication resources providing a first grade of service and communication resources providing a second grade of service inferior to the first grade of service, and wherein the communication resources providing the first grade of service can be converted into the communication resources providing the second grade of service and vice versa.

19. The resource allocator of claim 18, wherein the means for assigning the at least one communication resource to the reserved resource group further comprises means for selecting the at least one communication resource from the communication resources providing the first grade of service.

20. The resource allocator of claim 19, further comprising:

means for allocating a communication resource providing the second grade of service to the at least one queued service only when the reserved resource group is not depleted.

21. The resource allocator of claim 17, wherein the means for assigning the at least one other communication resource assigns the at least one other communication resource prior to allocating any of the plurality of communication resources to the at least one queued service.

22. The resource allocator of claim 17, wherein the means for assigning the at least one other communication resource further comprises:

means for providing the at least one other communication resource by performing at least one of: pre-empting usage of communication resources in an ongoing communication, re-allocating communication resources to the ongoing communication, and blocking one of the plurality of services.

23. The resource allocator of claim 17, further comprising:

means for releasing a deallocated resource to the plurality of communication resources when the reserved resource group is depleted; and

means for assigning yet another communication resource of the plurality of communication resources to the reserved resource group with greater preference relative to allocation of the plurality of communication resources to the at least one queued service.

24. A resource allocator for use in a wireless time division multiple access (TDMA) communication system comprising a plurality of periodically repeating time slots forming a plurality of communication resources, the wireless TDMA communication system supporting a plurality of services that include at least one queued service and at least one non-queued service, the resource allocator comprising:

means for assigning a first communication resource of the plurality of communication resources and providing a first grade of service to a reserved resource group used for servicing the at least one non-queued service;

means for allocating a communication resource from the reserved resource group to the at least one non-queued service; and

means for assigning, when the reserved resource group is depleted, at least one other communication resource of the plurality of communication resources and providing the first grade of service to the reserved resource group prior to allocating a second communication resource of the plurality of communication resources to the at least one queued service, wherein the second communication resource provides a second grade of service inferior to the first grade of service.

25. The resource allocator of claim 24, wherein the first grade of service corresponds to a first time slot interleave and the second grade of service corresponds to a second time slot interleave greater than the first time slot interleave.

26. The resource allocator of claim 24, further comprising:

means for allocating the second communication resource to the at least one queued service only when the reserved resource group is not depleted.

27. The resource allocator of claim 24, wherein the means for assigning the at least one other communication resource further comprises:

means for providing the at least one other communication resource by performing at least one of: pre-empting usage of communication resources in an ongoing communication, re-allocating communication resources to the ongoing communication, and blocking one of the plurality of services.

28. The method of claim 24, further comprising:

means for releasing a deallocated resource to the plurality of communication resources when the reserved resource group is depleted; and

means for assigning yet another communication resource of the plurality of communication resources and providing the first grade of service to the reserved resource group prior to allocating the second communication resource to the at least one queued service.

09/521,308

First Hit Fwd Refs

End of Result Set

Generate Collection Print

L23: Entry 6 of 6 File: USPT Nov 17, 1998

DOCUMENT-IDENTIFIER: US 5838968 A
TITLE: System and method for dynamic resource management across tasks in real-time operating systems

Abstract Text (1):
A system and method for dynamic resource management across tasks in real-time operating systems is disclosed. The system and method manage an arbitrary set of system resources and globally optimize resource allocation across system tasks in a dynamic fashion, according to a system specified performance model. The present invention provides a mechanism for system programmers to program tasks such that system performance will be globally optimized and dynamically managed over a system programmer-controllable set of system resources. The invention supports a mechanism for defining and managing arbitrary resources through a task resource utilization vector. Each task resource utilization vector contains an arbitrary number of task resource utilization records that contain quantities of system resources that each task qualitatively prefers to utilize while executing on the processor. Each of the task utilization records contains a run level that reflects the associated task's ability to perform its work when allocated the resources according to the particular task resource utilization record. This run level is used to dynamically vary the quantity of system resources that the task has allocated, based on the availability of system resources and the priorities of the tasks.

Brief Summary Text (4):
The present invention relates to operating systems, and more particularly to dynamic resource management between tasks in real time operating systems.

Brief Summary Text (8):
Real Time operating systems have been in use in a variety of embedded systems for many years. These systems were often special purpose systems such as operating systems to support embedded algorithms in military devices such as acquiring and tracking targets, or software to manage large switching systems. Real time operating systems all share similar restrictions. First the hardware available for processing is physically constrained, i.e. limited physical memory and backing store. Second, task reliability is essential. If a task fails due to lack of resources the entire system may cease to function. Hence resource allocation for real-time tasks must be precise and consistent.

Brief Summary Text (10):
In order for a real time task to meet its schedule or processing deadline, it must have access to the critical system resources necessary for processing. Previous systems attempt to manage resource constraints by simply checking once, usually at task initiation, whether there are enough resources available for the task to execute. Such systems may also provide rudimentary static allocation through a reservation system. In a reservation system, a task requests the resources it needs from the system, sometimes at task invocation, other times at system initialization. The system withholds those resources from the available pool for future use by the task. With static allocation, or reservation, of resources, resources can become fragmented or under-utilized easily. Additionally, the system

cannot readjust resources to a particular task once it has been allocated, or the other tasks may become resource-starved. Thus systems in which task needs or resource utilization may fluctuate, either because tasks' have terminated or because the tasks needs have changed, cannot readjust the resources available to other tasks. Static system resource optimization is thus limited to manual methods which can only approximate actual run-time optimal usage.

Brief Summary Text (11):
Systems are known which provide simple static memory reservation capabilities to ensure that tasks can receive the necessary amount of physical or virtual memory. Such systems include MacOS from Apple Computer Corporation, and UNIX originally developed by AT&T. These systems allow a task to register a necessary amount of memory and the operating system will not start the task unless enough free memory is available. These are examples of simple static resource allocation mechanisms. Because these systems only support static reservations, a task must reserve the maximum amount of resources necessary at any time during its execution in order to ensure adequate resources. This leads to under-utilization of the resource due to internal fragmentation. As the hardware components of systems have increased in complexity and memory has become less expensive, simple static allocation techniques no longer provide enough flexibility to optimize resource utilization. A new method capable of globally optimizing resource allocation for multiple resources across tasks is needed.

Brief Summary Text (12):
Globally optimizing and managing resource utilization provides many benefits to the real-time programmer. First the programmer is guaranteed that the task will not abnormally terminate due to the unavailability of resources. Second, the system can maximize the use of its resources based on the needs of the tasks currently operating, thus enabling more tasks to execute concurrently.

Brief Summary Text (13):
In order to understand the uses of dynamic resource allocation a brief discussion of real time operating system programming is in order. Real time systems are characterized by limited resources (hardware) and time criticality of operations. Real time systems are often embedded systems where the system cannot be readjusted or restarted easily. For example control processors in manufacturing environments, real time medical monitoring systems, or remote satellite imaging systems. Programmers of these systems are faced with resource limitations (hardware in satellites can't change), or time criticality constraints (patients may die if the monitoring software misses a hardware interrupt giving the system vital input). The unavailability of necessary resources, such as physical memory, main processor, or input/output bus cycles can cause problems in these environments. Therefore programmers must expend precious instruction cycles to check and verify that the resources they need are available or can be obtained.

Brief Summary Text (14):
This programming model is necessary because real time systems do not provide generalized support for resource re-allocation, at most they provide the static reservation capability discussed earlier. Often programmers must rely upon cooperative allocation of resources. In addition complex real time systems are usually programmed by more than one programmer, or may be later reprogrammed by the same programmer. In this case, the cooperative resource allocation mechanism is prone to error due to lack of enforceable allocation mechanisms and no re-allocation mechanisms.

Brief Summary Text (15):
Finally programmers must manually determine how to balance resources between high priority tasks and low priority tasks. If a low priority task requires a large number of resources, but is not time critical, programmers have little choice but to hope that those resources are available when the low priority task runs. In

static reservation systems a low priority task cannot pre-allocate all the resources it needs since that would make those resources unavailable for time critical tasks. Yet if a time critical task fails to deallocate its resources, the low priority task may never be able to execute since it cannot acquire the resources necessary to run.

Brief Summary Text (17):
The present invention provides a new method for optimal resource management between tasks in a real time multitasking environment. It is one aspect of the present invention to provide a resource allocation mechanism for multiple resources. It is another aspect of the invention to provide for dynamic resource management facilitating migration of resources from one task to another and from resource to resource. It is a further aspect of this invention that the allocation of resources is globally optimized and dynamically managed across all tasks in the real time operating system.

Brief Summary Text (18):
The present invention provides a mechanism for system programmer to program tasks such that system performance will be globally optimized and dynamically managed over a system programmer-controllable set of system resources. The invention supports a mechanism for defining and managing arbitrary resources through a task resource utilization vector. Each task utilization vector contains an arbitrary number of task resource utilization records that contain quantities of the pluralities of system resources that each of the plurality of tasks qualitatively prefers to utilize while executing on the processor. Each of the these task utilization records contains a run level reflecting the associated task's ability to perform its work when allocated the resources according to the particular task resource utilization record. This run level is used to dynamically vary the quantity of the plurality of system resources that tasks have allocated based on the availability system resources and the priorities of the plurality of tasks.

Brief Summary Text (19):
The invention also provides for a flexible global optimization mechanism that enables the resource manager to dynamically manage system resources. The optimization mechanism operates by keeping track of actual system resource utilization through periodic measuring by updating the current task utilization record to reflect the consumption of the of the plurality of system resources, and by using this information to allocate or deallocate resources from tasks in order to satisfy system resource requests.

Brief Summary Text (20):
Tasks participate in the resource management by responding to queries from the system to determine if they can change their consumption of at least one of the system resources. Next the global optimizer computes a plurality of global system performance scores based on the responses to the queries of the tasks and a specified system performance model.

Brief Summary Text (21):
Finally the invention provides for setting each of the tasks to the resource utilization record associated with the global system performance score which optimizes system performance according to the specified system performance model. In the present invention the system performance model takes into account the priorities of the tasks, the resource utilization run level associated with the particular task and the priority of the task requests to determine the allocation or deallocation of system resources.

Brief Summary Text (22):
One advantage of this invention is that it dynamically manages system resources. Another advantage of the invention is that the type of system resources managed are not limited. System programmers may specify the types of systems resources they

wish to use on a task by task basis with a plurality of resource usage levels, not just a static allocation. A system programmer may add their own resources to be managed into the system resource master tables, thus enabling the operating system to dynamically manage resource contention on their behalf.

Brief Summary Text (23):
Another advantage of the invention is that the system performance model is capable of utilizing the various different run levels the task has specified in order to globally (across all system resources) optimize system performance.

Brief Summary Text (24):
By providing a reservation system for programmers the present invention minimizes error due to unavailability of resources, and relieves the programmer from reliance upon weak cooperative resource allocation mechanisms. In addition, by providing dynamic resource re-allocation, the present invention provides for improved overall system utilization and optimization. Finally, the present invention allows for resource cooperation between time critical high priority tasks and low priority tasks. The invention and its advantages will be better understood by referring to the description of the preferred embodiment and the figures.

Drawing Description Text (2):
FIG. 1 is a block diagram of a processing system utilizing the media engine multimedia system in accordance with the present invention

Drawing Description Text (3):
FIG. 2 is a block diagram of the master resource list utilized by the resource manager of FIG. 1.

Drawing Description Text (5):
FIG. 4 is block diagram of the task list which is managed by the resource manager of FIG. 1.

Drawing Description Text (7):
FIG. 5 is a block diagram of the degradation method executed by the resource manager of FIG. 1.

Drawing Description Text (8):
FIG. 6 is a block diagram of the promotion method executed by the resource manager of FIG. 1.

Detailed Description Text (15):
Referring now to FIG. 1, a block diagram of an embodiment of a multimedia system 10 in accordance with the present invention is shown. Multimedia system 10, is a complete multimedia processing system. It includes host system 15, media engine subsystem 25, and input/output and multimedia (I/O) subsystem 35. Multimedia system 10 also includes an operating system which includes resource manager 170 and the real time operating system, XOS 180; the operating system interacts with, manages, and controls the various elements of multimedia system 10. More specifically, resource manager 170 controls static resource allocation and I/O bandwidth allocation of various components in media engine subsystem 25, such as memory 110 allocation, media processor 20 processor cycles, and I/O bandwidth to memory 110 through memory bus 109. XOS 180 is the real time operating system that manages the tasks that run on media processor 20. XOS 180 provides support functions for media processor 20 tasks such as interrupt processing, context switching, and subroutine calling.

Detailed Description Text (16):
Host system 15 includes host processor 120, disk drive 121, and memory 123. Host processor 120 is coupled to disk drive 121, and memory 122. Host processor 120 executes the resource manager 170. In the preferred embodiment, host processor 120

conforms to the Intel X86 architecture, but those skilled in the art will recognize that the host processor could be adapted to other architectures such as the PowerPC.TM. architecture, and other processor architectures.

Detailed Description Text (17):
Media engine subsystem 25 is connected to host system 15 through PCI bus 40. Media engine subsystem 25 contains a memory 110 which is accessed by media processor 20 through memory bus 109, which is controlled by memory control circuit 50. PCI bus 40 is an industry standard bus that couples memory controller 50 with host processor 120 of host system 15. Memory controller 50 controls the access to memory 110 and mitigates access from media processor 20, and host processor 120 over PCI bus 40. Memory 110 is a dynamic memory from which XOS 180 executes. XOS 180 and resource manager 170 also use memory 110 to transfer data from devices and to store and access communication subsystems implemented between XOS 180 and resource manager 170. In the preferred embodiment, memory 110 is dynamic memory, a Rambus DRAM, but those skilled in the art will recognize that other types of memory such as static memory would operate equally well.

Detailed Description Text (21):
Multimedia I/O subsystem 35 contains a plurality of devices for sending and receiving input and output. Multimedia I/O subsystem includes direct device access 80, high bandwidth device 140, Pbus 90, and output device 150. In the present embodiment direct device access 80 is a buffer coupled to register file 30 to which high bandwidth devices 140 may be attached. Direct device access 80 is coupled to and allows high bandwidth device 140 to send and receive data directly to or from a portion of the register file. High bandwidth devices 140 may be any of a variety of devices including special purpose optical devices, such as cameras, or recording devices, or display device.

Detailed Description Text (22):
In the preferred embodiment host processor 120 is a processor running the Windows.TM. or MS-DOS.TM. operating system. However those skilled in the art will recognize that host processor 120 could be any type of processor and in fact in some embodiments, the real time operating system XOS 180 and the resource manager 170 could be operating on a common processor.

Detailed Description Text (23):
Real time tasks execute under XOS 180 on the media engine chip, while resource manager 170 is responsible for creating and dynamically managing the resources available to tasks. Tasks have a task structure commonly known in the art as a task control block that contains the control information necessary for a task to run, be suspended and resumed.

Detailed Description Text (24):
Resource manager 170 controls resource allocation and distribution for all XOS 180 tasks. resource manager 170 is responsible for globally maximizing resource utilization across all XOS 180 tasks. In order to perform its resource optimization functions, resource manager 170 maintains current allocation status information for each resource it manages. This information is stored in a master list which is periodically updated by resource manager 170.

Detailed Description Text (25):
Referring to FIG. 2, Resource master list 200 is coupled to host processor 120 and resides in host memory 123, however in another embodiment resource master list 200 could be located on memory 110, or in any other resource manager 170 accessible location. Resource master list 200 holds the state of all resources managed by resource manager 170. For each resource managed by resource manager 170, a resource entry 210 is created. Resource Entry 210 contains the resource indicator, 220 which could be a name or an index, the maximum number of allocable units, 230, and the currently allocated units 240. In the present embodiment, resources include media

engine CPU 20 utilization, PCI bus 40 utilization, memory 100 utilization, and memory bus 109 utilization, and host processor 120 utilization. However it should be noted that the list of resources is not exhaustive. System programmers may add their own resources as resource master list 200 is designed to be expandable. Examples of other types of resources include special purpose hardware and other input/output devices. The maximum number of allocable units is established by the hardware configuration and set by the system at startup time.

Detailed Description Text (27):
Resource manager 170 periodically updates the usage values of global system resources in resource master list 200 by calling an update routine when a timer event occurs. This facilitates task creation by maintaining current information based on actual resource usage, thus ensuring the maximum number of concurrent tasks will be supported. The routine GetPlatformUsage allows resource manager 170 to obtain recent information about actual resource usage.

Detailed Description Text (28):
In the present embodiment, each task is responsible for deciding the amount of each resource it requires. Referring to FIG. 3, task 350 can specify any number of resource configurations in task resource utilization vector 300. Task resource utilization vectors are located in host memory 123 and are linked together in the host memory. In the present embodiment, task resource utilization vector 300 is coupled to task 350. Tasks must specify at least one task resource utilization record, 310, which specifies the required quantities of the resources managed by resource manager 170 that are necessary for Task 350 to function properly. Task resource utilization record 310, contains an index, 315, indicating the run level associated with this record, and multiple entries 311,312, 314. Each entry initially contains the quantity of the resource requested at this particular level. If a resource entry has no quantity associated with it, i.e. 0, resource manager 170 assumes the task has no requirement for that resource. This does not mean that the task does not use the resource, simply that it does not have a quantifiable need for the resource.

Detailed Description Text (29):
Those skilled in the art will recognize that task resource utilization vector 300 need not reside within the task structure itself, but could be an independent structure located within either resource manager 170 or XOS 180 so long as task 350, and resource manager 170 can access the vector.

Detailed Description Text (30):
In addition to the required first task resource utilization record, a task may specify multiple task resource utilization records. Task resource utilization records are sorted in preferred order. The first record, 310, is the most favorable allocation of resources that task 350 could use. Additional records specify other resource utilization configurations. Resource utilization configurations may be in an order. The last record specifies the minimal resource utilization configuration indicating that a resource utilization configuration below this will cause the task to fail due to resource constraints. It is not necessary that resource utilization configurations are monotonicaly decreasing for each resource. Rather they are qualitatively decreasing indicating the task will function less well at a lower level. Thus the highest level could specify 250 KB of memory and 1% CPU, and the next lower level could specify 100 KB of memory and 5% CPU indicating that the task can perform with less memory at the cost of greater CPU consumption, but will still not perform as well as if the entire 250 KB were available.

Detailed Description Text (31):
In the present embodiment tasks have three classes, error intolerant, error-tolerant realtime, and non-realtime. To guarantee proper functioning of error intolerant tasks, the resource manager must reserve resources for the worst-case usage scenario of these tasks.

Detailed Description Text (32):
Referring to FIG. 3, task 350 has 3 task resource utilization records, 310, 320, and 330. Record 310 contains a list of resources that task 350 would like to have available while executing. If task 350 cannot be allocated the resources specified in 310, task 350 could execute and perform its functions with the resources specified in record 320 or 330. However, if the resources specified in record 330 are not available, then task 350 could not execute and perform its functions properly.

Detailed Description Text (34):
When a task is operating using a specified task resource utilization record configuration, this is called the run level. Run levels can by symbolic, e.g. optimal, optimal, next, minimal, or numeric, 1-n. This creates a matrix of run level, resource requirements which resource manager 170 uses to globally optimize resource utilization across all tasks. In the present embodiment, task resources associated with the current run level are updated with actual resource usage measurements.

Detailed Description Text (36):
If the media engine subsystem becomes resource constrained, and tasks have difficulty gaining access to needed resources then resource manager 170 must decide whether to lower the available resources for current tasks, or fail the task allocation request. The act of retrieving resources from an existing task is called degradation. Degradation occurs when a task is asked to give up some of its resources and move to a lower run level.

Detailed Description Text (37):
In the present embodiment, when resource degradation is necessary, tasks are degraded, i.e. the resource requirements are lowered, following system performance model:

Detailed Description Text (39):
2) Tasks with equal priority will be degraded equally, i.e. an attempt will be made to lower their resource requirements by the same amount; and

Detailed Description Text (40):
3) An optimal configuration will be sought, where optimal means degrading total system resources as little as possible, without violating rules 1) and 2), while maintaining real-time requirements.

Detailed Description Text (43):
A newly allocated task requires more resources than are currently available

Detailed Description Text (44):
Media engine errors (such as missed deadlines) cause the resource manager 170 to trigger resource degradation

Detailed Description Text (46):
Degradation may be performed under two different task execution environments: when tasks run with unequal priority and when tasks run with equal priority. In both situations, resource manager 170 seeks a global optimal configuration, one in which it degrades the total system resources as little as possible according to the rules of the task selection algorithm while still providing the needed resources.

Detailed Description Text (48):
According to FIG. 4, Resource manager 170 maintains a task list sorted in priority order, lowest to highest. In the present embodiment, the list is a set of vectors with each vector representing all the tasks in the system at that priority level. System task list 400 resides in host processor memory 123, and contains a list of

tasks sorted by priority order. At each priority level, there may be one or more tasks currently instantiated. Priority vector 410 shows tasks 405 and 406 both operating at priority level T0. Those skilled in the art will recognize that the vector of tasks at each priority level may be implemented using a variety of structures including, but not limited to, doubly linked lists, object classes, arrays, and other structures.

Detailed Description Text (51):
The degradation process, QueryResult, is executed by resource manager 170 on host processor 120. In another embodiment the degradation process could be executed on XOS 180. Referring to FIG. 5, In step 500, the deficit to be filled is determined. A resource deficit can be created by any of the triggering conditions: a new task is going to be created; a task misses a deadline; or an existing task requests more resources.

Detailed Description Text (52):
In step 510, the process first checks the currently available amount of resources. The routine GetPlatformUsage provides the necessary information.

Detailed Description Text (53):
If the requested resources are available, as testing in step 520, then the request is fulfilled and the resources are allocated in step 530. If the resources are not available the resource manager 170, executing step 540, will scan through task list 400 and perform two functions.

Detailed Description Text (56):
1. The task can indicate to resource manager 170 to use the information in my task resource vector. So for example for task 350, which is operating at record 320, it could be lowered to the utilization levels in resource utilization record 330.

Detailed Description Text (59):
Next in 550, resource manager 170 computes an optimal system utilization score. In the present embodiment, this computation enforces the degradation performance model. Thus a score is computed taking into account:

Detailed Description Text (61):
(2) Tasks with equal priority will be degraded equally, i.e. an attempt will be made to lower their resource requirements by the same amount. In the present embodiment this means equivalently lowering tasks at the same priority level by proportional run levels.

Detailed Description Text (62):
In addition, if the priority of the requesting task is lower than that of the task being examined for the purposes of computing the optimal system utilization score, the task will not be included in the computation unless the task in step 540 volunteered to give up resources.

Detailed Description Text (63):
In the present embodiment, the optimal system utilization configuration is computed by iterating through all tasks at all run levels and computing the system utilization score within the above parameters. In other embodiments the algorithm for computing the system utilization score could be based on particular physical constraints, or on task class types. Those skilled in the art will recognize many different algorithms may be used.

Detailed Description Text (64):
It is worth noting in that in the present embodiment, different run levels for tasks which are being considered need not be monotonicaly decreasing in resource usage. Thus in one run level, memory 100 use actually may increase in order to counteract decreases in other resources. If both memory 100 use and CPU use are

high, then QueryResult may chose complementary resources. For instance, it may increase memory 100 access for one task to reduce its CPU requirement. It may reduce another task's memory 100 use at the expense of increasing that task's CPU requirement. By manipulating two or more variables in one or more task's resource usage, the resource manager 170 balances and solves the tasks' contention for resources.

Detailed Description Text (65):
In step 560, the resources recovered are compared against the deficit. If there are now enough resources available, QueryResult tells the tasks to set themselves to the specified run levels computed in the optimal system utilization level, step 580, and the process executes step 530 to allocation the resources. Otherwise in step 570, the request for additional resources is denied.

Detailed Description Text (66):
In addition to degrading tasks, resource manager 170 seeks to increase resources available to tasks by promoting tasks.

Detailed Description Text (67):
Promotion occurs when a task is allowed to consume more resources and move to a higher run level. Promoting a task increases a task's performance by allocating more resources to it when necessary. The resource manager 170 promotes a task under the following conditions:

Detailed Description Text (69):
resource manager 170 finds resources during one of its periodic resource checks and decides to promote a task

Detailed Description Text (70):
Resources become available due to task termination or task freeing resources.

Detailed Description Text (71):
Promotion follows the same general model as degradation. In fact the same routine QueryResult is used. Referring now to FIG. 6. The process of promotion is discussed. In step 600, the surplus to be filled is determined. A resource surplus can be created by any of the triggering conditions: a task terminates; resource manager 170 finds resources; or an existing task requests more resources.

Detailed Description Text (72):
In step 610, the process first checks the currently available amount of resources. The routine GetPlatformUsage provides the necessary information.

Detailed Description Text (73):
If the requested resources are available, as testing in step 620, then the request is fulfilled and the resources are allocated in step 630. If the resources are not available the resource manager 170, executing step 640, will scan through task list 400 and perform two functions.

Detailed Description Text (76):
1. The task can indicate to resource manager 170 to use the information in my task resource vector. So for example for task 350, which is operating at record 320, it could be raised to the utilization levels in resource utilization record 310.

Detailed Description Text (79):
Next in step 650, resource manager 170 computes an optimal system utilization score. In the present embodiment, this computation enforces the degradation performance model. Thus a score is computed taking into account:

Detailed Description Text (81):
(2) Tasks with equal priority will be promoted equally, i.e. an attempt will be

made to raise their resource requirements by the same amount. In the present embodiment this means equivalently raising tasks at the same priority level by proportional run levels.

Detailed Description Text (82):
In the present embodiment, the optimal system utilization configuration is computed by iterating through all tasks at all run levels and computing the system utilization score within the above parameters. In other embodiments the algorithm for computing the system utilization score could be based on particular physical constraints, or on task class types. Those skilled in the art will recognize many different algorithms may be used.

Detailed Description Text (83):
It is worth noting in that in the present embodiment, different run levels for tasks which are being considered need not be monotonicaly decreasing in resource usage. Thus in one run level, memory 100 use actually may increase in order to counteract decreases in other resources. If both memory 100 use and CPU use are high, then QueryResult may chose complementary resources. For instance, it may increase memory 100 access for one task to reduce its CPU requirement. It may reduce another task's memory 100 use at the expense of increasing that task's CPU requirement. By manipulating two or more variables in one or more task's resource usage, the resource manager 170 balances and solves the tasks' contention for resources.

Detailed Description Text (84):
In step 660, the resources recovered are compared against the deficit. If there are now enough resources available, QueryResult tells the tasks to set themselves to the specified run levels computed in the optimal system utilization level, step 680, and the process executes step 630 to allocation the resources. Otherwise in step 670, the request for additional resources is denied.

Detailed Description Paragraph Table (2):
TABLE 1 RMThreadAttr
UpdateResourceMeasurement
RMPlatformManager::UpdateResourceMeasurement virtual void UpdateResourceMeasurement ( ) Effects: Periodically synchronizes and updates the resource manager's current measurement of resource consumption. Requires Resources such as the Media engine's CPU cycle consumption : require periodic monitoring, and the resource manager's cached estimates of these resources remain as accurate as possible. The resource manager or a timer message issued by the RM's VxD calls UpdateResourceMeasurement ( ). Modifies Internal RM structures. : Returns: Nothing.

Detailed Description Paragraph Table (3):
QueryResult handler(DWORD ref,
RMAdviseCommand command, RMUsageInfo *usageInfo, const RMUsageInfo& delta)
Effects: Implements commands defined in command structure including degradation usageInfo pointer to an RMUsageInfo block delta: for degradation, the values in this resource vector represent the current resource deficit/resources needed. For promotion, the values represents current free resources available. (Note: in the case of degradation, a resource with a negative deficit can be ignored - there is no conflct for that resource . In the case of promotion, no resource value should be set negative

CLAIMS:

1. A method of dynamic resource management on a data processing system including a processor, at least one system resource, and a plurality of tasks, the processor being coupled to the system resource and capable of executing the plurality of tasks, the method comprising the steps of:

executing instructions on the processor to create at least two tasks capable of executing on the processor, each of the at least two tasks having a priority;

creating at least one task resource utilization vector for each of the at least two tasks, the task resource utilization vector comprising the quantity of the at least one system resource that each of the at least two tasks prefers to utilize while executing on the processor; and

dynamically varying the quantity of the at least one system resource that the at least two tasks have allocated based on the availability of the at least one system resource and the priorities of the at least two tasks.

2. The method of claim 1 wherein said task resource utilization vector comprises a plurality of task resource utilization records and the steps of dynamically varying the quantity of the at least one system resource include:

for each task resource utilization record of the plurality, assigning a run level to each task utilization record reflecting the associated task's ability to perform its work when allocated the resources according to each task resource utilization record;

initially allocating each of the at least two tasks an associated task utilization record such that the task with the higher priority obtains the quantity of the at least one system resource specified in the associated task utilization record which guarantees the most efficient performance of the higher priority task;

measuring the actual at least one system resource utilization of the at least two tasks on a periodic basis;

and adjusting the at least two tasks to use the at least one system resource such that the at least two tasks execute use of the plurality of task utilization records based on the efficient use of the data processing system.

3. The method of claim 2 where determining the efficient use of the data processing system comprises the steps of:

for each of the at least two tasks computing the current consumption of the at least one system resource;

updating the current task utilization record to reflect the consumption of the at least one system resource;

querying the at least two tasks to determine if they can change their system resource consumption;

computing a plurality of global system performance scores based on the responses to the queries of the at least two tasks and a specified system performance model; and

setting each of the at least two tasks to the resource utilization record associated with the global system performance score which optimizes system performance according to the specified system performance model.

4. A method of dynamic resource management on a data processing system including a processor, a plurality of system resources, and a plurality of tasks, the processor being coupled to the system resource and capable of supporting the tasks, the method comprising the steps of:

executing instructions on the processor to create a plurality of tasks capable of

executing on the processor, each of the plurality having a priority;

creating a plurality of task resource utilization records for each of the plurality of tasks, each task resource utilization record of the plurality comprising quantities of the pluralities of system resources that each of the plurality of tasks qualitatively prefers to utilize while executing on the processor;

for each task resource utilization record of the plurality, assigning a run level to the task utilization record reflecting the associated task's ability to perform its work when allocated the resources according to each task resource utilization record; and

dynamically varying the quantity of the plurality of system resources that the plurality of tasks have allocated based on the availability of the plurality of system resources and the priorities of the plurality of tasks.

5. The method of claim 4 where the dynamic varying of the allocation of the plurality of system resources comprises:

periodically measuring the actual system resource utilization of the plurality of tasks;

updating the plurality of task resource utilization records to reflect the consumption of the of the plurality of system resources;

querying each of the plurality of tasks to determine if they can change their consumption of at least one of the plurality of system resources;

computing a plurality of global system performance scores based on the responses to the queries of the plurality of tasks and a specified system performance model; and

setting each of the plurality of tasks to the run level associated with the global system performance score which optimizes system performance according to the specified system performance model.

6. A system for dynamic resource management on a data processing system including a plurality of processors, a plurality of system resources, and a plurality of tasks, the plurality of processors being coupled to the plurality of system resources and capable of supporting the tasks, the system comprising:

instructions that when executed on at least one of the plurality of processors create a plurality of tasks capable of execution on at least one of the plurality of processors, each of the plurality of tasks having a priority;

means for creating a plurality of task resource utilization records for each of the plurality of tasks, each task resource utilization record of the plurality comprising quantities of the pluralities of system resources that each of the plurality of tasks qualitatively prefers to utilize while executing on the at least one processor;

means for assigning a run level to each of the task utilization records of the plurality of tasks, each run level reflecting the associated task's ability to perform its work when allocated the resources according to each task resource utilization record; and

means for dynamically varying the quantity of the plurality of system resources that the plurality of tasks have allocated based on the availability of the plurality of system resources and the priorities of the plurality of tasks.

7. A method of dynamic resource management on a data processing system including a processor and a system resource coupled to the processor, the method comprising the steps of:

executing instructions on the processor to create a plurality of tasks capable of execution on the processor, each of the tasks having a priority;

creating a task resource utilization vector for each of the plurality of tasks, the task resource utilization vector including at least one task utilization resource record specifying a resource quantity request; and

dynamically varying the resource quantity requests for each of the plurality of tasks based on the availability of the system resource and the priorities of at least two of the plurality of tasks.

10. The method of claim 7 wherein at least one task resource utilization vector includes a plurality of task resource utilization records, the step of dynamically varying the resource quantity requests further comprising:

assigning a run level to each task resource utilization record, each run level reflecting the associated task's ability to perform its work when allocated resources according to the associated task's resource utilization record;

allocating the resource to each task based on each task's resource utilization records, run level, and priority, such that a first task having a priority greater than that of a second task will receive a first portion of the resource according to a preferred run level while a second portion of the resource is allocated to the second task;

measuring actual resource utilization of each task on a periodic basis; and

adjusting at least one of the plurality of tasks' resource utilization records such that the execution of the plurality of tasks is based on the efficient use of the resource.

13. The method of claim 10 where adjusting at least one of the plurality of tasks' resource utilization records further comprises the steps of:

computing the consumption of the resource for at least one task;

updating at least one task utilization record to reflect consumption of the resource;

querying at least one task to determine if it can change its resource consumption;

computing a plurality of global system performance scores based on the response to querying at least one task and a specified system performance model; and

setting at least one task to a resource utilization record associated with the global system performance score which optimizes system performance according to the specified system performance model.

14. A method of dynamic resource management on a data processing system including a plurality of processors and a system resource coupled to the plurality of processors, the method comprising the steps of:

executing instructions on at least one of the plurality of processors to create a plurality of tasks capable of execution on any of the plurality of processors, each of the tasks having a priority;

creating a task resource utilization vector for each of the plurality of tasks, the task resource utilization vector including at least one task utilization resource record specifying a resource quantity request; and

dynamically varying the resource quantity requests for each of the plurality of tasks based on the availability of the system resource and the priorities of at least two of the plurality of tasks.

15. The method of claim 14 wherein at least one task resource utilization vector includes a plurality of task resource utilization records, the step of dynamically varying the resource quantity requests further comprising:

assigning a run level to each task resource utilization record, each run level reflecting the associated task's ability to perform its work when allocated resources according to the associated task's resource utilization record;

allocating the resource to each task based on each task's resource utilization records and priority, such that a first task having a priority greater than that of a second task will receive a first portion of the resource according to a preferred run level while a second portion of the resource is allocated to the second task;

measuring actual resource utilization of each task on a periodic basis; and

adjusting at least one of the plurality of task's resource utilization records such that the execution of the plurality of tasks is based on the efficient use of the resource.

16. The method of claim 15 where adjusting at least one of the plurality of tasks' resource utilization records further comprises the steps of:

computing the consumption of the resource for at least one task;

updating at least one task utilization record to reflect consumption of the resource;

querying at least one task to determine if it can change its resource consumption;

computing a plurality of global system performance scores based on the response to querying at least one task and a specified system performance model; and

setting at least one task to a resource utilization record associated with the global system performance score which optimizes system performance according to the specified system performance model.

17. A computer system comprising:

a plurality of resources including a processor and a memory coupled to the processor; and

a resource manager program, the program being executed by the processor and including:

a routine to create a plurality of tasks capable of execution on the processor, each of the tasks having a priority;

a routine to create a task resource utilization vector for each of the plurality of tasks, the task resource utilization vector including at least one task utilization resource record specifying a resource quantity request; and

a routine to dynamically varying the resource quantity requests for each of the

plurality of tasks based on the availability of the system resource and the priorities of at least two of the plurality of tasks.

19. The computer system of claim 17 wherein at least one task resource utilization vector includes a plurality of task resource utilization records, and the routine to dynamically varying the resource quantity requests further comprises:

a run level assignment routine, the run level assignment routine assigning a run level to each task resource utilization record, each run level reflecting the associated task's ability to perform its work when allocated resources according to the associated task's resource utilization record;

an allocation routine, the allocation routine allocating the resource to each task based on each task's resource utilization records, run level, and priority, such that a first task having a priority greater than that of a second task will receive a first portion of the resource according to a preferred run level while a second portion of the resource is allocated to the second task;

a resource measurement routine measuring actual resource utilization of each task on a periodic basis; and

an adjustment routine, the adjustment routine adjusting at least one of the plurality of tasks' resource utilization records such that the execution of the plurality of tasks is based on the efficient use of the resource.

20. The computer system of claim 19 wherein the adjustment routine further comprises:

a resource consumption routine, the resource consumption routine computing the consumption of the resource for at least one task;

an update routine, the update routine updating at least one task utilization record to reflect consumption of the resource;

a query routine querying at least one task to determine if it can change its resource consumption;

a performance score routine, the performance score routine computing a plurality of global system performance scores based on the response to querying at least one task and a specified system performance model; and

a setting routine, the setting routine setting at least one task to a resource utilization record associated with the global system performance score which optimizes system performance according to the specified system performance model.

22. The computer program product of claim 21 wherein at least one task resource utilization vector includes a plurality of task resource utilization records, and the routine to dynamically varying the resource quantity requests further comprises:

a run level assignment routine, the run level assignment routine assigning a run level to each task resource utilization record, each run level reflecting the associated task's ability to perform its work when allocated resources according to the associated task's resource utilization record;

an allocation routine, the allocation routine allocating the resource to each task based on each task's resource utilization records, run level, and priority, such that a first task having a priority greater than that of a second task will receive a first portion of the resource according to a preferred run level while a second portion of the resource is allocated to the second task;

a resource measurement routine measuring actual resource utilization of each task on a periodic basis; and

an adjustment routine, the adjustment routine adjusting at least one of the plurality of tasks' resource utilization records such that the execution of the plurality of tasks is based on the efficient use of the resource.

23. The computer program product of claim 22 wherein the adjustment routine further comprises:

a resource consumption routine, the resource consumption routine computing the consumption of the resource for at least one task;

an update routine, the update routine updating at least one task utilization record to reflect consumption of the resource;

a query routine querying at least one task to determine if it can change its resource consumption;

a performance score routine, the performance score routine computing a plurality of global system performance scores based on the response to querying at least one task and a specified system performance model; and

a setting routine, the setting routine setting at least one task to a resource utilization record associated with the global system performance score which optimizes system performance according to the specified system performance model.